OK2440 用户手册 V1.0

保定飞凌嵌入式技术有限公司

网站：http://www.witech.com.cn

论坛：http://www.witech.com.cn/bbs

目　录

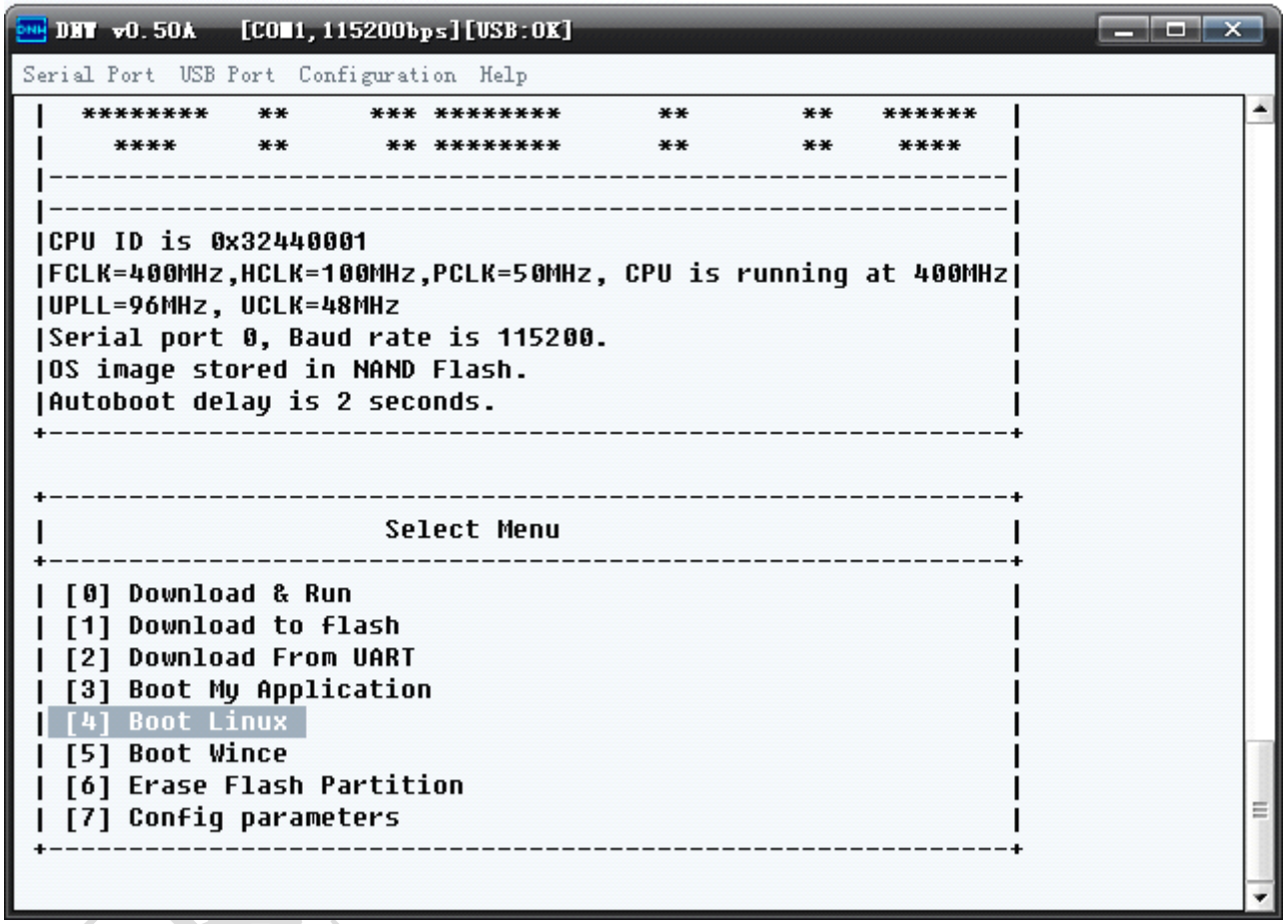
一. 预装 LINUX 使用说明.....4
1.1 如何启动 LINUX.....4
1.2 设置 LINUX 上电自启动.....5
1.3 LINUX 的参数设置.....6
1.4 文件系统的选择.....6
1.5 鼠标与触摸屏的切换.....6
1.6 通过串口与 PC 传送文件.....8
1.6.1 向 PC 机传送文件.....8
1.6.2 接收 PC 机传过来的文件.....8
1.7 挂接 NFS 网络文件系统.....10
1.8 挂接 SD 卡.....12
1.9 挂接 U 盘.....12
1.10 CMOS 摄像头(OV9650)的使用.....14
1.11 USB 摄像头的使用.....15
1.12 设置并保存系统时钟.....18
1.13 测试音频 录音与播放.....19
1.14 测试网络连接.....20
1.14.1 查看网络设置.....20
1.14.2 重新设置 IP.....20
1.14.3 使用 PING 命令.....21
1.14.4 如何 DHCP 动态获取 IP 和 DNS.....22
1.15 WEBSERVER 功能.....24
1.15.1 启动 WebServer.....24
1.15.2 浏览 Web 服务器上的网页.....25
1.16 使用 TELNET 远程登录开发板.....26
1.17 使用 FTP 传递文件.....27
1.18 如何开机自动运行程序.....28
1.19 如何中止程序运行.....29
二. LINUX 开发教程.....31
2.1 如何安装 REDHAT 9.0.....31
2.2 在 REDHAT 中添加新用户.....41
2.3 建立交叉编译环境.....42
2.4 编译内核.....43
2.5 内核配置说明.....45
2.5.1 配置 CPU.....46
2.5.2 配置 USB 鼠标和键盘.....47
2.5.3 配置 U 盘支持.....48
2.5.4 配置 SD/MMC 卡驱动.....51
2.5.5 配置串口驱动.....52
2.5.6 配置声卡驱动.....53
2.5.7 如何配置触摸屏.....56
2.5.8 如何配置网眼和中芯微等 USB 摄像头.....59
2.5.9 如何配置 LED 驱动.....61

2.5.10 如何配置 RTC 实时时钟驱动 62
2.5.11 如何配置 yaffs 文件系统的支持 64
2.5.12 配置文件系统（EXT2/EXT3/FAT/NFS） 66
2.5.13 配置 CS8900 网卡驱动 71
2.6 CRAMFS 文件系统制作 73
2.7 配置 PC 机 LINUX 的 FTP 服务 74
2.8 配置 PC 机 LINUX 的 TELNET 74
2.9 编译应用程序 75
2.10 把应用程序传到目标板上运行 76
2.11 应用程序实验 77
2.11.1 串口编程 77
2.11.2 网络编程 78
2.11.3 线程编程 81
2.12 驱动程序实验 82
2.12.1 LED 驱动实验 83
2.12.2 AD 驱动实验 88
三. LINUX 图形编程 94
3.1 用户界面 GUI 介绍 94
3.1.1 常用 GUI 介绍 95
3.1.2 关于 Qt 96
3.1.3 Qt/Embedded 简介 98
3.1.4 Qtopia 介绍 99
3.2 QT 常用工具的介绍 101
3.2.1 Qt 设计器（Qt Designer） 101
3.3.2 Tmake 104
3.3 QT 的编程 106
3.3.1 准备实验嵌入式图形开发环境的搭建 106
3.3.2 准备安装软件包 106
3.3.3 建立 QTE 开发环境所需要的目录 106
3.3.4 各个目录功能介绍 107
3.3.5 编译 QTOPIA 所依赖的库文件 107
3.3.6 交叉编译 qtopia 111
3.3.7 将 qtopia 挂在到目标板上 113
3.3.8 简单的“hello world”实验 115
附录一 MINICOM 使用指南 120
附录二 LINUX 常用命令详解 126
附录三内核配置详细说明 131

一.预装 Linux 使用说明

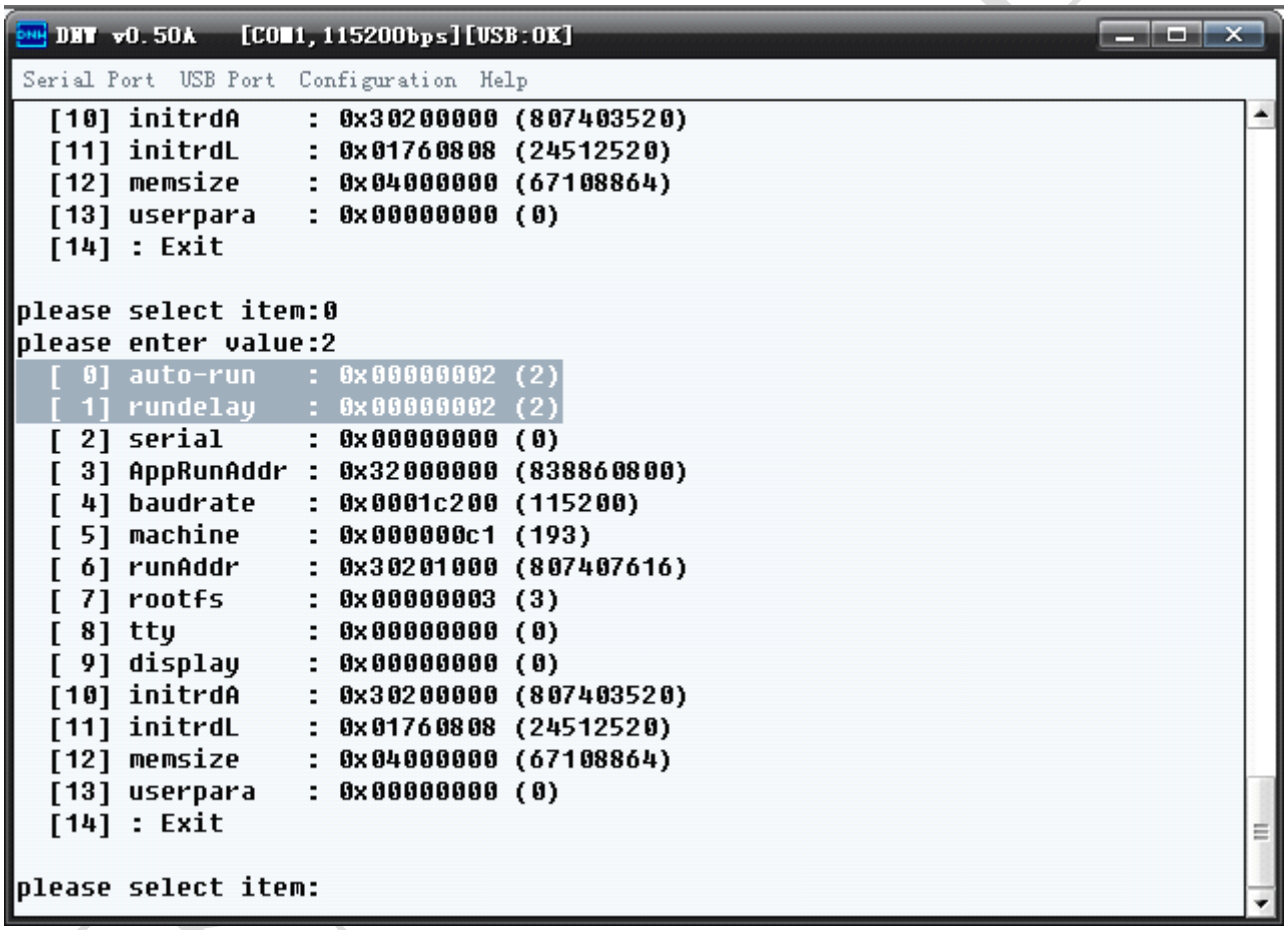
1.1 如何启动 linux

OK2440 开发板出厂时预装了 wince 操作系统和 linux 操作系统。给板子上电后在调试终端 DNW 上按任意键进入 bootloader 菜单，选择 4 将启动 linux 操作系统。



1.2 设置 linux 上电自启动

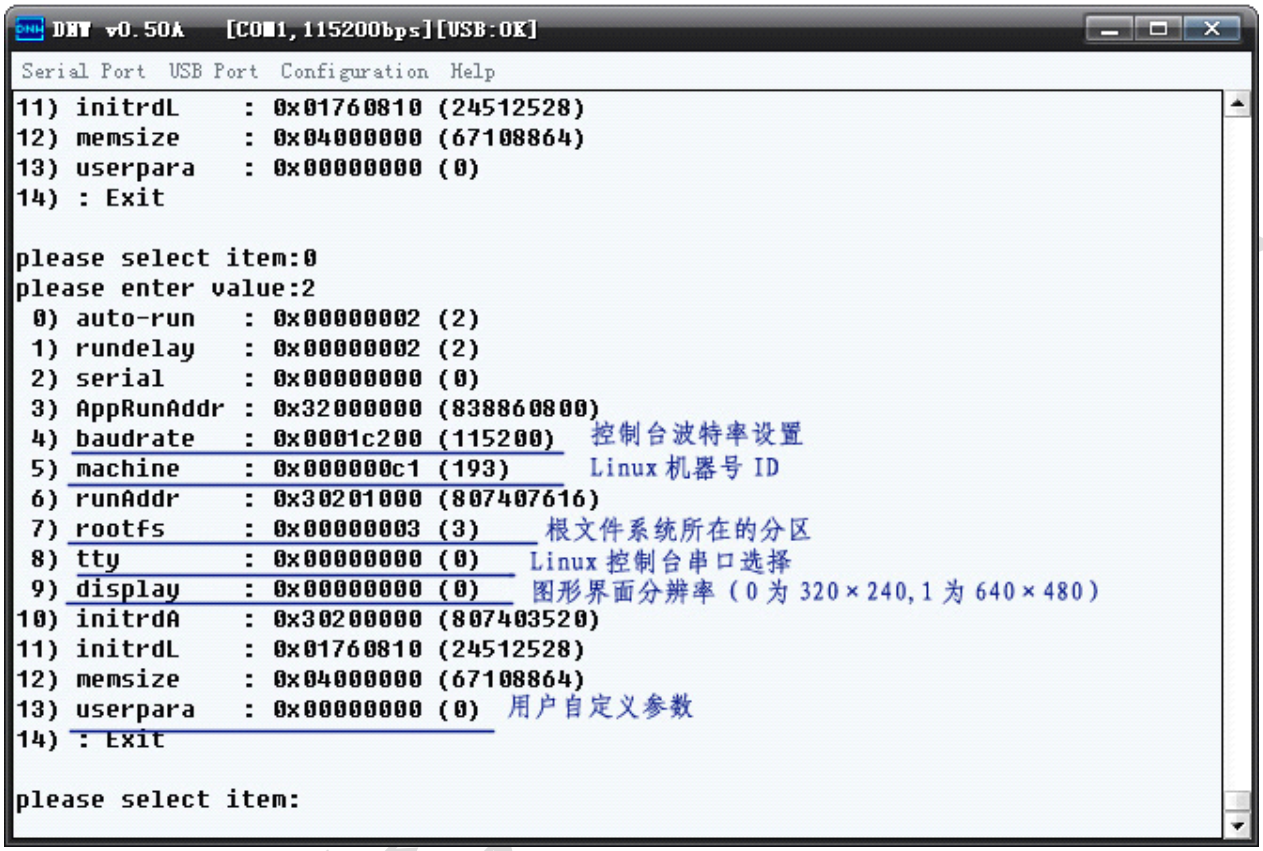
上电后进入 bootloader 菜单，选择 7 进入参数设置菜单，将参数[0]的值改为 2（1 为自启动自己的应用程序即分区【application】中的程序，2 为自启动 linux，3 为自启动 wince)。将参数[1]的值改成非零值（延时大小)。



修改完毕后，退出并保存。然后给板子复位或重新上电，则板子会自动进入 linux 系统。

1.3 linux 的参数设置

bootloader 中关于 linux 的参数是：4、5、7、8、9、13。如图：



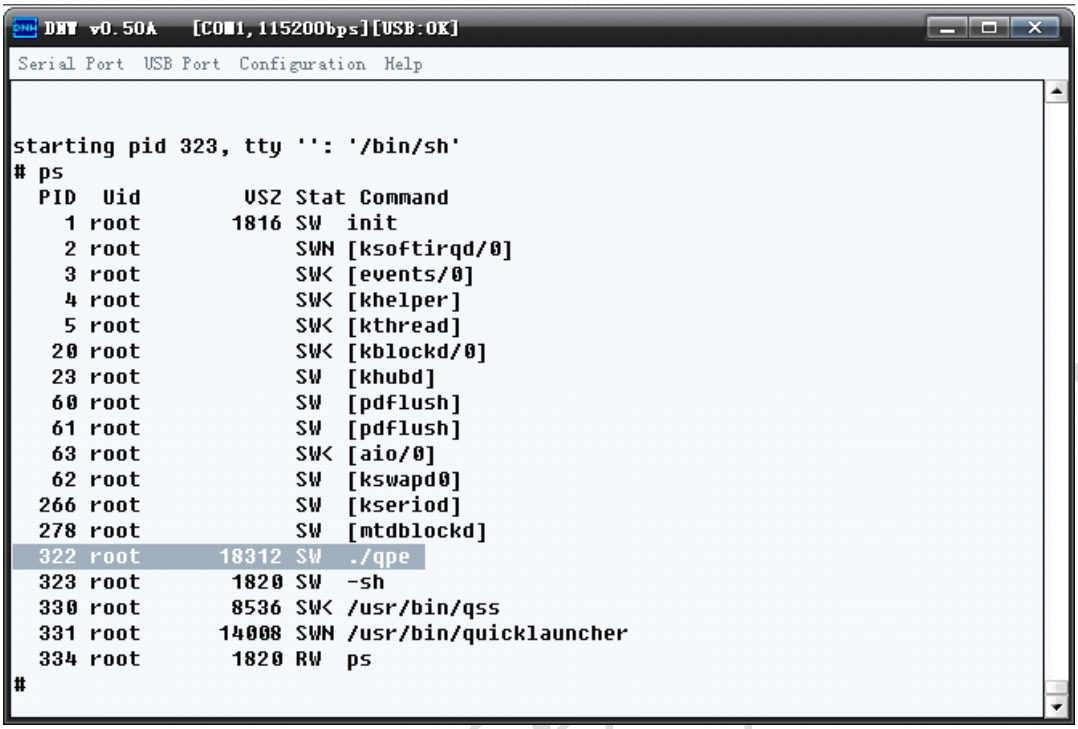
1.4 文件系统的选择

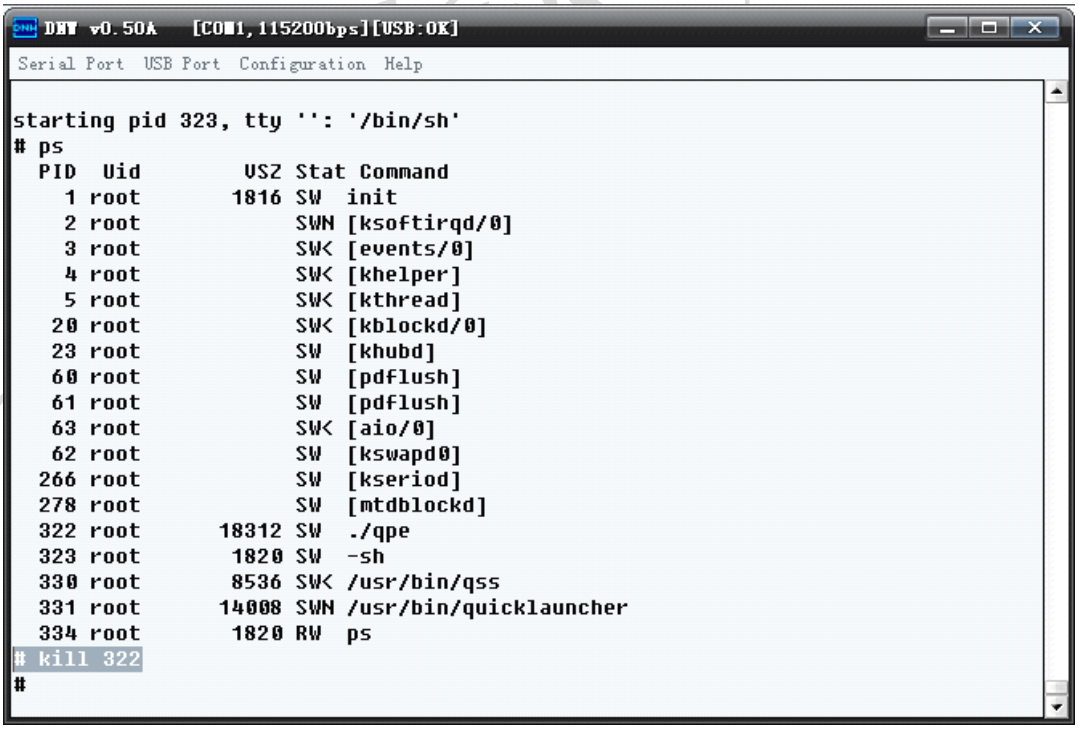
OK2440V2 开发板支持 cramfs 和 yaffs 两种格式的文件系统，由于 flash 空间问题，预装的文件系统中，cramfs 文件系统包含了图形文件系统，yaffs 只做了一个根文件系统。通过修改 bootloader 的参数 7 的值来选择所要启动的文件系统。3 为 cramfs 文件系统，4 为 yaffs 文件系统。

1.5 鼠标与触摸屏的切换

qte 中，鼠标和触摸屏不能同时使用，预装的一般是支持鼠标的。由于 qte 是在后台运行的，所以在切换到鼠标与触摸支持前需要先杀死原来的进程：输入 ps 命令可查看当前进程，找到 qpe 的进程号，用 kill 命令杀死该进程。

然后进入/usr/etc 目录可看到 rc.mouse 和 rc.touch，运行 rc.touch 可支持触摸屏：./rc.touch；由触摸切换到鼠标支持同理。


DNW v0.50A [COM1,115200bps][USB:OK]
Serial Port USB Port Configuration Help
starting pid 323, tty '': '/bin/sh'
ps
PID Uid VSZ Stat Command
1 root 1816 SW init
2 root SWN [ksoftirqd/0]
3 root SW< [events/0]
4 root SW< [khelper]
5 root SW< [kthread]
20 root SW< [kblockd/0]
23 root SW [khubd]
60 root SW [pdflush]
61 root SW [pdflush]
63 root SW< [aio/0]
62 root SW [kswapd0]
266 root SW [kseriod]
278 root SW [mtdblockd]
322 root 18312 SW ./qpe
323 root 1820 SW -sh
330 root 8536 SW< /usr/bin/qss
331 root 14008 SWN /usr/bin/quicklauncher
334 root 1820 RW ps
#


DNW v0.50A [COM1,115200bps][USB:OK]
Serial Port USB Port Configuration Help
starting pid 323, tty '': '/bin/sh'
ps
PID Uid VSZ Stat Command
1 root 1816 SW init
2 root SWN [ksoftirqd/0]
3 root SW< [events/0]
4 root SW< [khelper]
5 root SW< [kthread]
20 root SW< [kblockd/0]
23 root SW [khubd]
60 root SW [pdflush]
61 root SW [pdflush]
63 root SW< [aio/0]
62 root SW [kswapd0]
266 root SW [kseriod]
278 root SW [mtdblockd]
322 root 18312 SW ./qpe
323 root 1820 SW -sh
330 root 8536 SW< /usr/bin/qss
331 root 14008 SWN /usr/bin/quicklauncher
334 root 1820 RW ps
kill 322
#

cd /usr/etc

./rc.touch

1.6 通过串口与 PC 传送文件

linux 下用串口与 PC 传送文件需要用超级终端。启动 linux 后关掉 DNW，打开超级终端并配置（配置方法详见使用基础部分）。按回车键进入 linux 命令行。

1.6.1 向 PC 机传送文件

点击超级终端菜单栏　发送—>接收文件，或在超级终端的空白处右击，在下拉菜单中选择接收文件。

设置要存放文件的目录，点关闭。

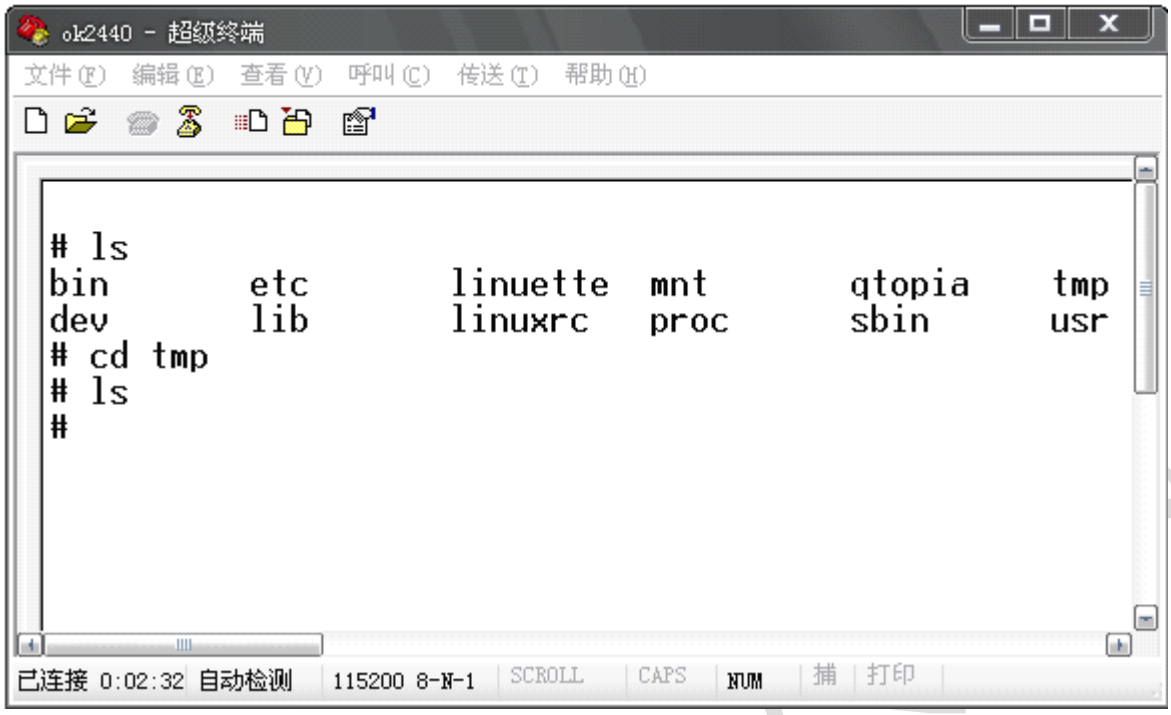
在命令行上输入：

```
sz /usr/etc/rc.local
```

可以看到在接收目录里出现了 rc.local 文件，该文件是 linux 启动时的配置文件。

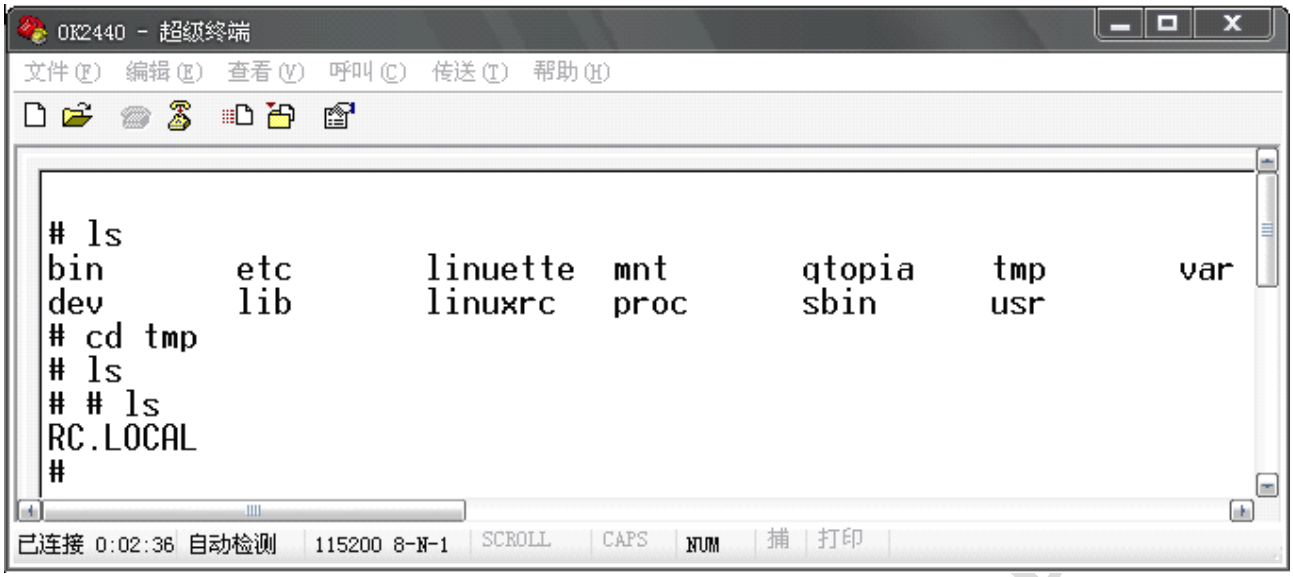
1.6.2 接收 PC 机传过来的文件

在 linux 命令行下进入（cd）要存放文件的目录，



现在我们把刚才传过来的文件在传到 linux 根目录下的 tmp 中。在超级终端上点右键—>发送文件，选择要发送的文件，点击发送。

传送成功后即可在命令行下看到该文件了。



一般传过来的文件是不可执行的，可以通过以下命令增加运行权限：

chmod +x 文件名

注：只读文件夹不能接收 PC 机传过来的文件，一般可传到/tmp 文件夹下。

1.7 挂接 NFS 网络文件系统

网络文件系统（NFS）是一种在网络上的机器间共享文件的方法，文件就如同位于客户的本地硬盘驱动器上一样。Red Hat Linux 既可以是 NFS 服务器也可以是 NFS 客户，这意味着它可以把文件系统导出给其它系统，也可以挂载从其它机器上导入的文件系统。

1. 为什么使用网络文件系统

NFS 对于在同一网络上的多个用户间共享目录很有用途。譬如，一组致力于同一工程项目的用户可以通过使用 NFS 文件系统（通常被称作 NFS 共享）中的一个挂载为 /myproject 的共享目录来存取该工程项目的文件。要存取共享的文件，用户进入各自机器上的 /myproject 目录。这种方法既不用输入口令又不用记忆特殊命令，就仿佛该目录位于用户的本地机器上一样。

打个比方，有甲乙两台机子，则甲机可以通过挂载网络文件系统来共享乙机的文件。

2. 挂载 NFS

挂接网络文件系统需要我们 PC 机端的操作系统是 LINUX 操作系统，建议用 redhat9.0

1、用交叉网线将 PC 机与开发板连接起来

2、在 PC 机上修改/etc/exports 文件，在里面写入“/ *”保存(注意/ *之间有空格)

注：/etc/exports 为 NFS 服务器的配置文件，“/ *”是将根目录下所有文件挂载

3、在桌面上点击右键->新建终端 输入 service nfs restart 以重新启动 nfs 服务

注：service nfs restart —— 重启 nfs 服务

service nfs start —— 启动 nfs 服务

service nfs stop —— 停止 nfs 服务

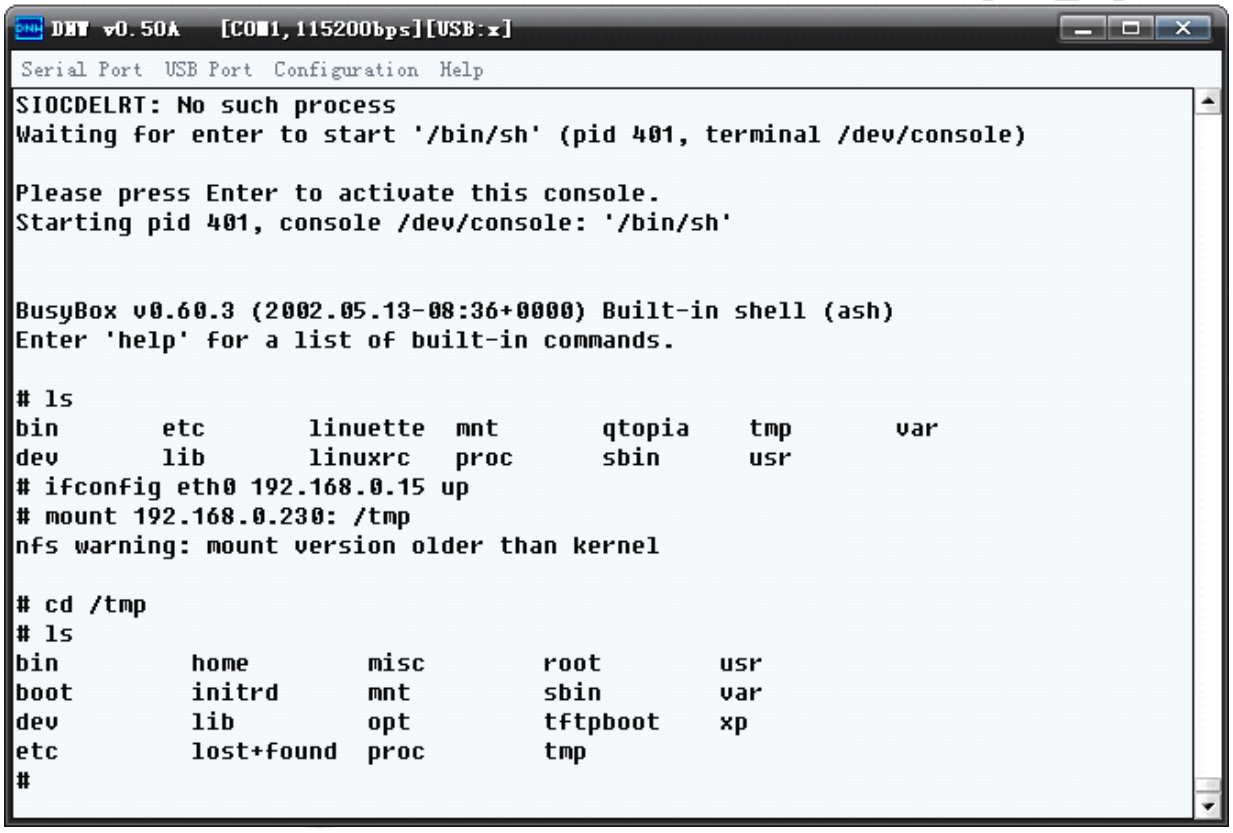
4、在目标板的 linux 操作系统下输入以下命令：

ifconfig eth0 192.168.0.15 up （该命令用于设置开发板的 IP 地址，可以通过 ifconfig 查看 IP）

mount 192.168.0.230: /tmp （192.168.0.230 为主机的 IP 地址）

也可以通过下边的命令挂载

mount -n -o nolock,rsize=1024,wsize=1024,timeo=15 192.168.0.230:/ /tmp



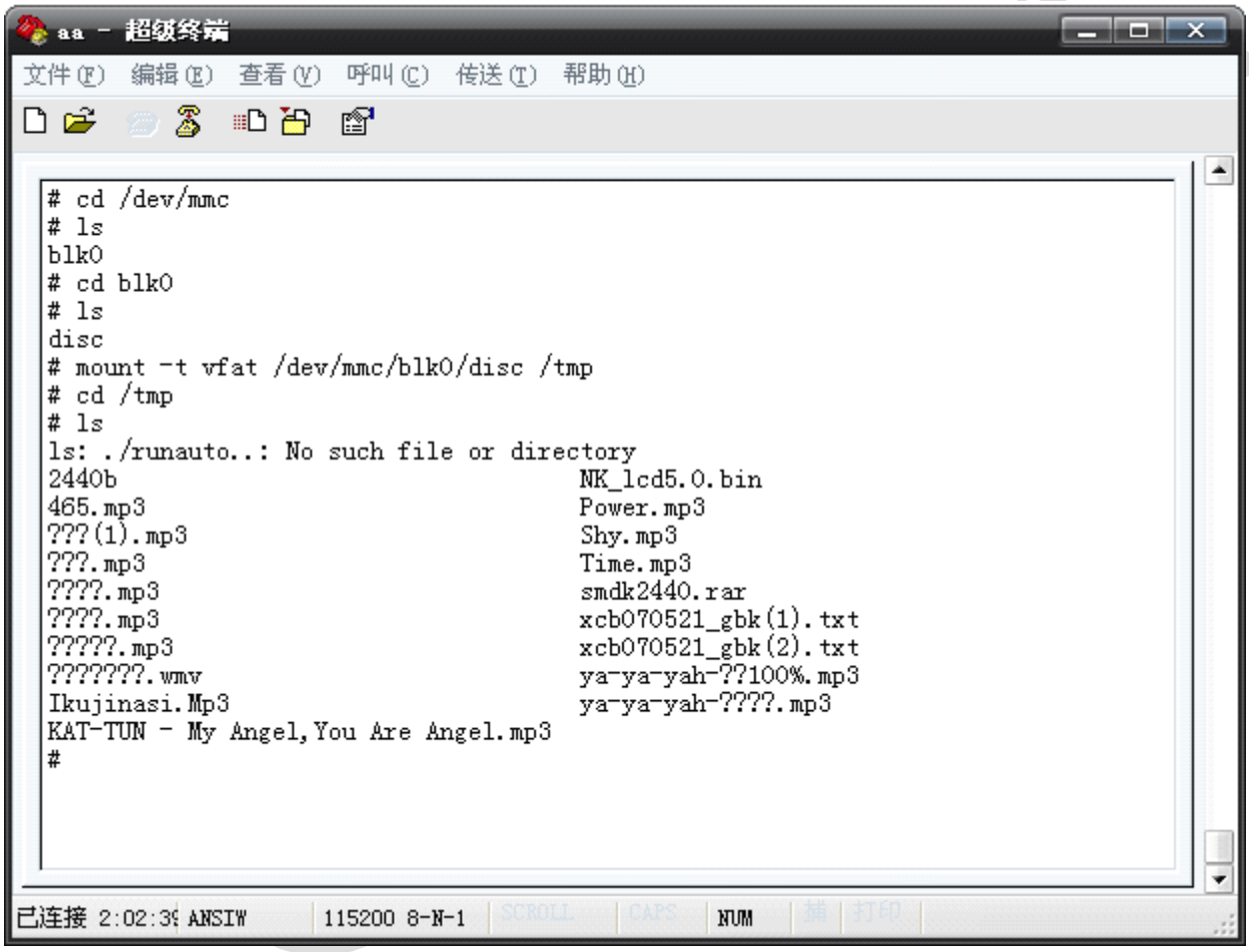
这样就把 PC 机的整个文件系统 mount 到目标板上了。以后我们做的应用程序经过交叉编译后可以直接在开发板上找到该程序并运行，以便调试修改，等程序完善后再将其集成到文件系统里。文件系统的制作参考 6.6 章节。

1.8 挂接 SD 卡

启动前插上 SD 卡，进入 liunx 命令行，可以看到/dev/mmc/blk0 目录下有一个 disc，该设备就是 SD 卡，通过以下命令挂接：

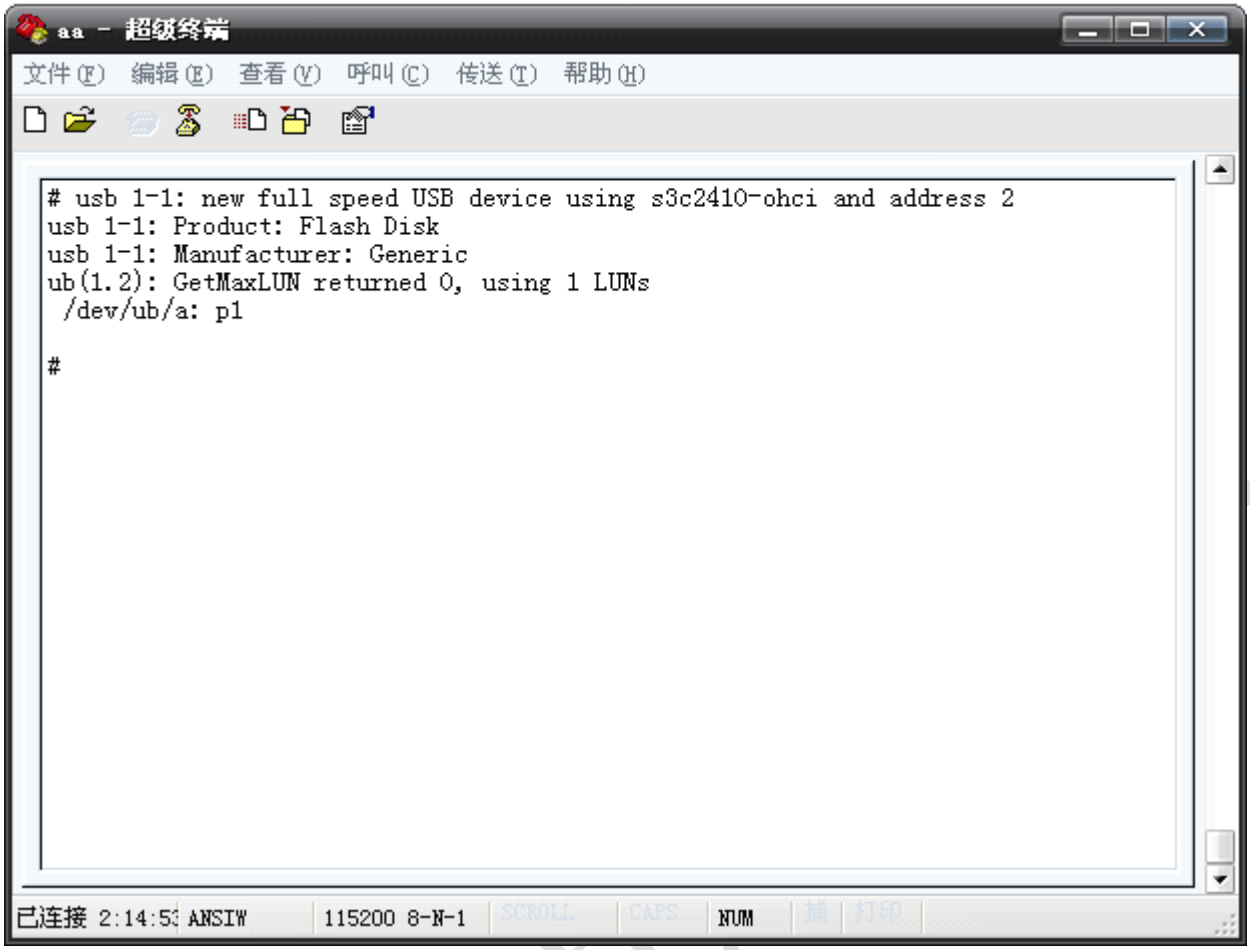
```
mount  -t vfat /dev/mmc/blk0/disc /tmp
```

挂接成功后可以在/tmp 目录下看到 SD 上的文件



1.9 挂接 U 盘

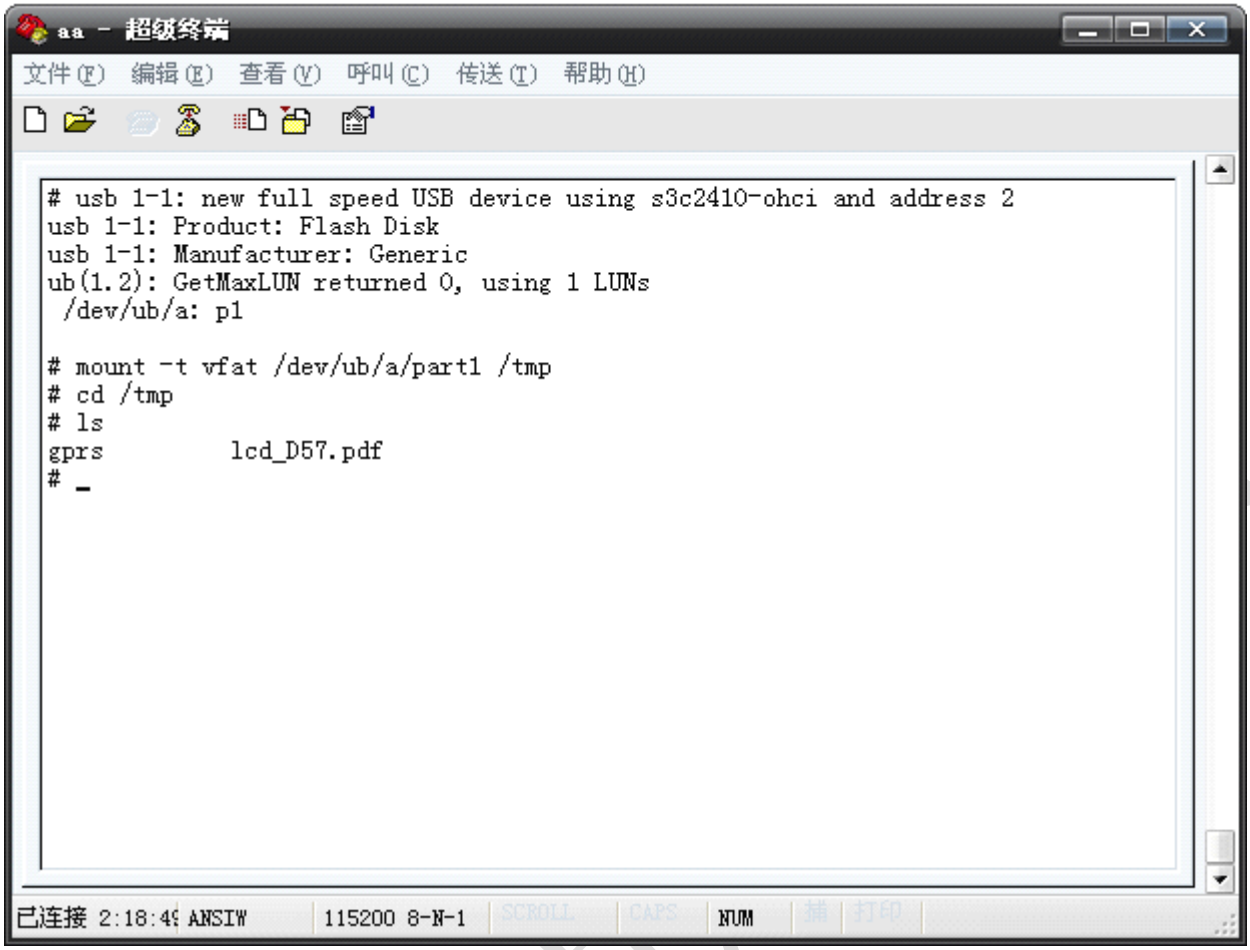
linux 启动后插上 U 盘，终端上出现以下打印信息：


aa - 超级终端
文件(F) 编辑(E) 查看(V) 呼叫(C) 传送(T) 帮助(H)
usb 1-1: new full speed USB device using s3c2410-ohci and address 2
usb 1-1: Product: Flash Disk
usb 1-1: Manufacturer: Generic
ub(1.2): GetMaxLUN returned 0, using 1 LUNs
/dev/ub/a: p1
#
已连接 2:14:5 ANSIW 115200 8-N-1 SCROLL CAPS NUM 捕 打印

/dev/ub/a 目录下的 part1 即为 U 盘设备文件，通过以下命令挂接

```
mount  -t vfat /dev/ub/a/part1 /tmp
```

挂接成功后可在/tmp 目录下看到 U 盘上的文件。



可通过 umount 命令卸载：

umount /tmp

1.10 CMOS 摄像头(OV9650)的使用

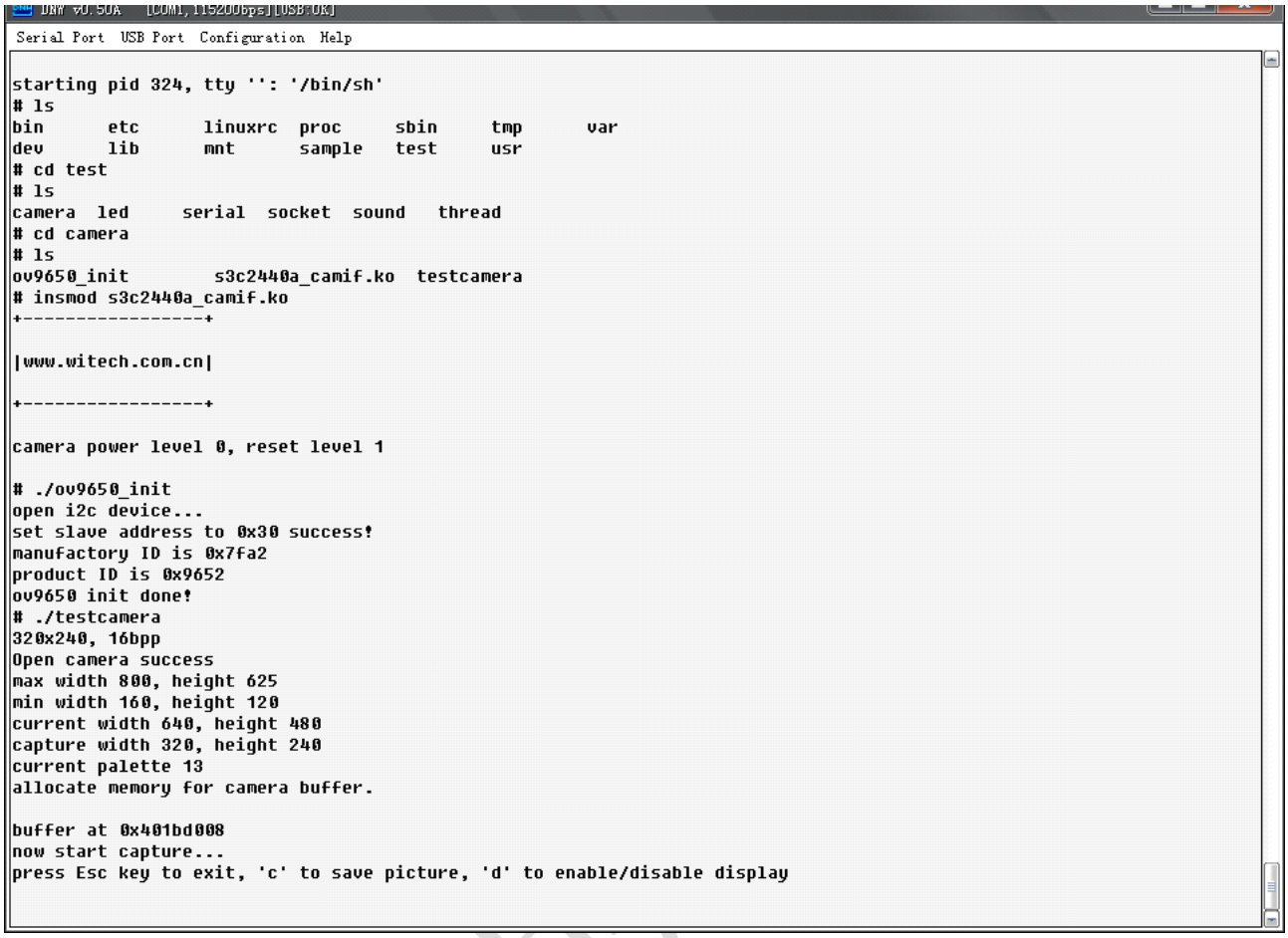
本开发板不仅支持下边提到的 USB 摄像头，还支持 CMOS（芯片为 OV9650）摄像头。首先把摄像头插好，其后上电，启动 linux 操作系统，进入根目录下的 test-〉camera，最后操作如下：

1)。加载驱动模块：insmod s3c2440a_camif.ko (设备节点为：/dev/v4l/video，驱动相关文件在光盘 linux 文件夹下面的驱动文件夹下)

2)。运行程序：./ov9650_init (通过 iic 初始化 ov9650 芯片,运行一次即可)

3)。运行测试程序：./testcamera (加-h 可查看参数说明)

如下图：



注：

1）按 C 保存图片，但本文件系统为只读的，用户需将相关文件拷贝到 tmp 下运行，图片保存在当前目录；

2）驱动模块中不含摄像头芯片的 iic 初始化，初始化部分位于 ov9650_init 中。

3）该模块的源码暂时只提供给 OEM/ODM 用户。

1.11USB 摄像头的使用

OK2440 开发板支持市场上比较常见的中芯微芯片的 USB 摄像头，把摄像头插入 USB 接口后，一般会弹出如下信息，同时/dev 目录下出现相应的设备名：/dev/v4l/video0。在命令行中输入如下命令：

ln –s /dev/v4l/video0 /dev/video0

```
drivers/usb/media/spca5xx/spca_core.c: USB SPCA5XX camera found. Type
Vimicro Zc301P 0x301b


# cd /dev/v4l
# ls
video0
#
```

1.本地液晶显示

这时在命令行中输入“serv-local”提示信息如下，此时本地液晶上就会显示摄像头的图像。

```
# serv-local
320x240, 16bpp
 servfox version: 1.1.3 date: 11:12:2005 (C) mxhaard@magic.fr
wrong spca5xx device
videoIn.framesizeIn = 19200
```

2.远程监控

本开发板不仅支持本地液晶显示还可以通过网络实现远程监控。在此之前要先安装一个软件 Java Web Start(在光盘中有，双击就可以安装了),之后的具体操作如下：

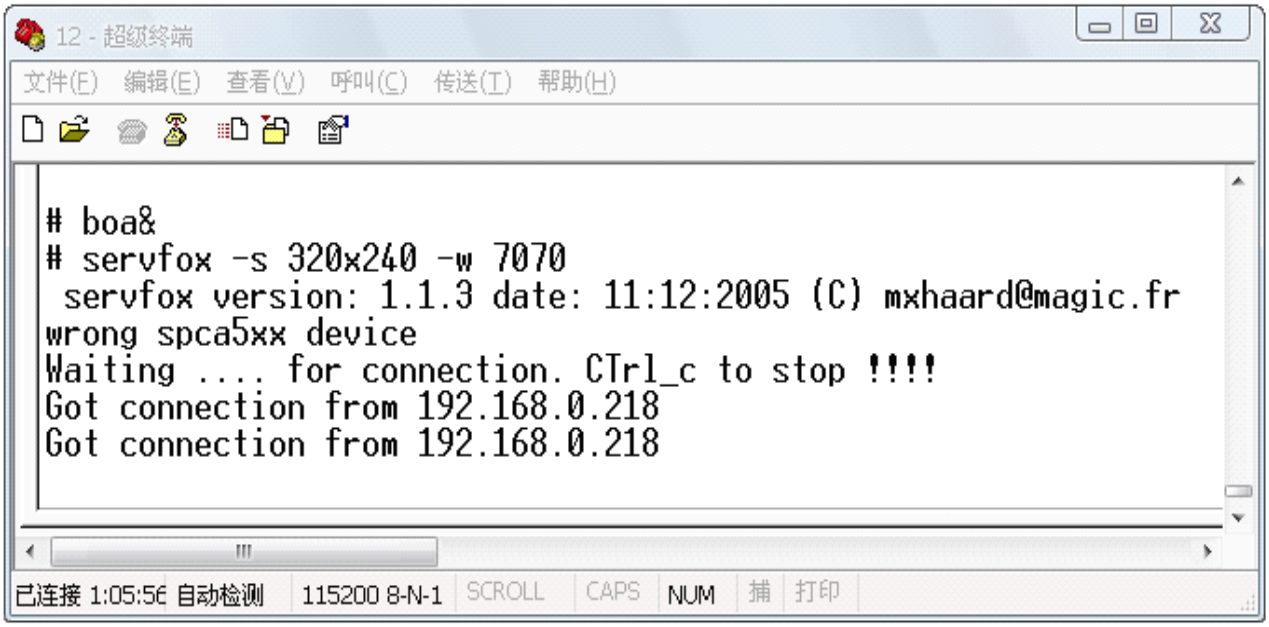
（这一切是在板子已经启动并插好摄像头之后操作的）

在板子的命令行输入如下命令：

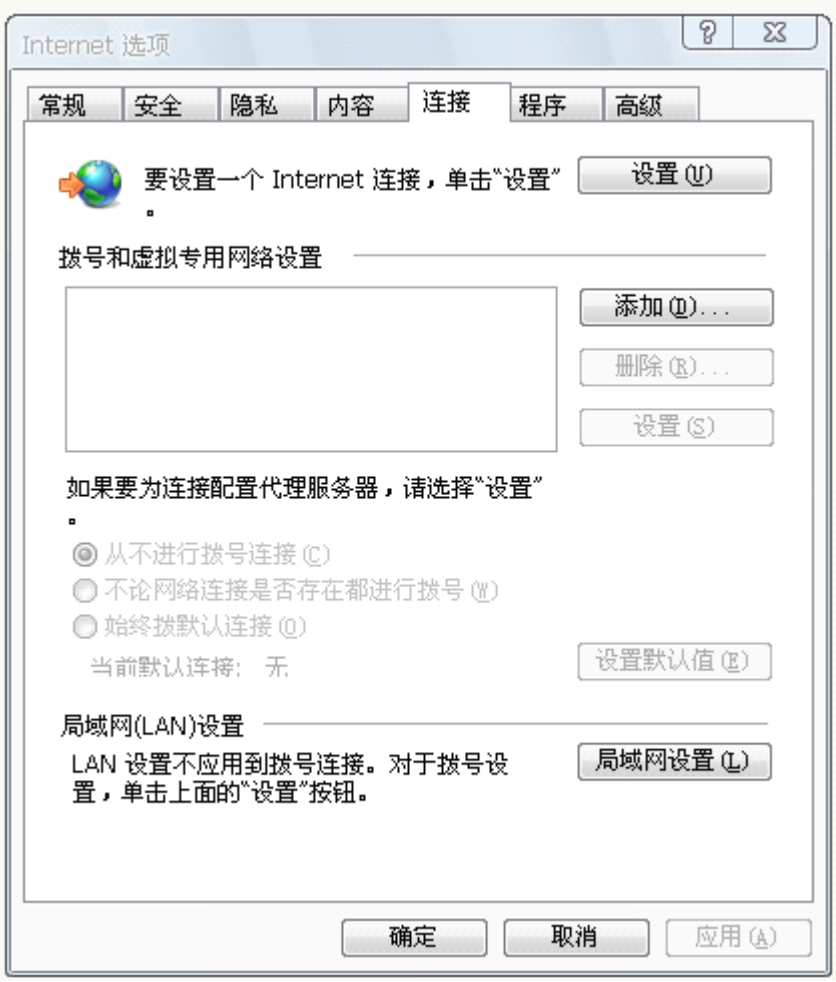
boa&

servfox -g –s 320x240 –w 7070

(注意分辨率 320x240 不能写成 320*240，那是个字母 x)

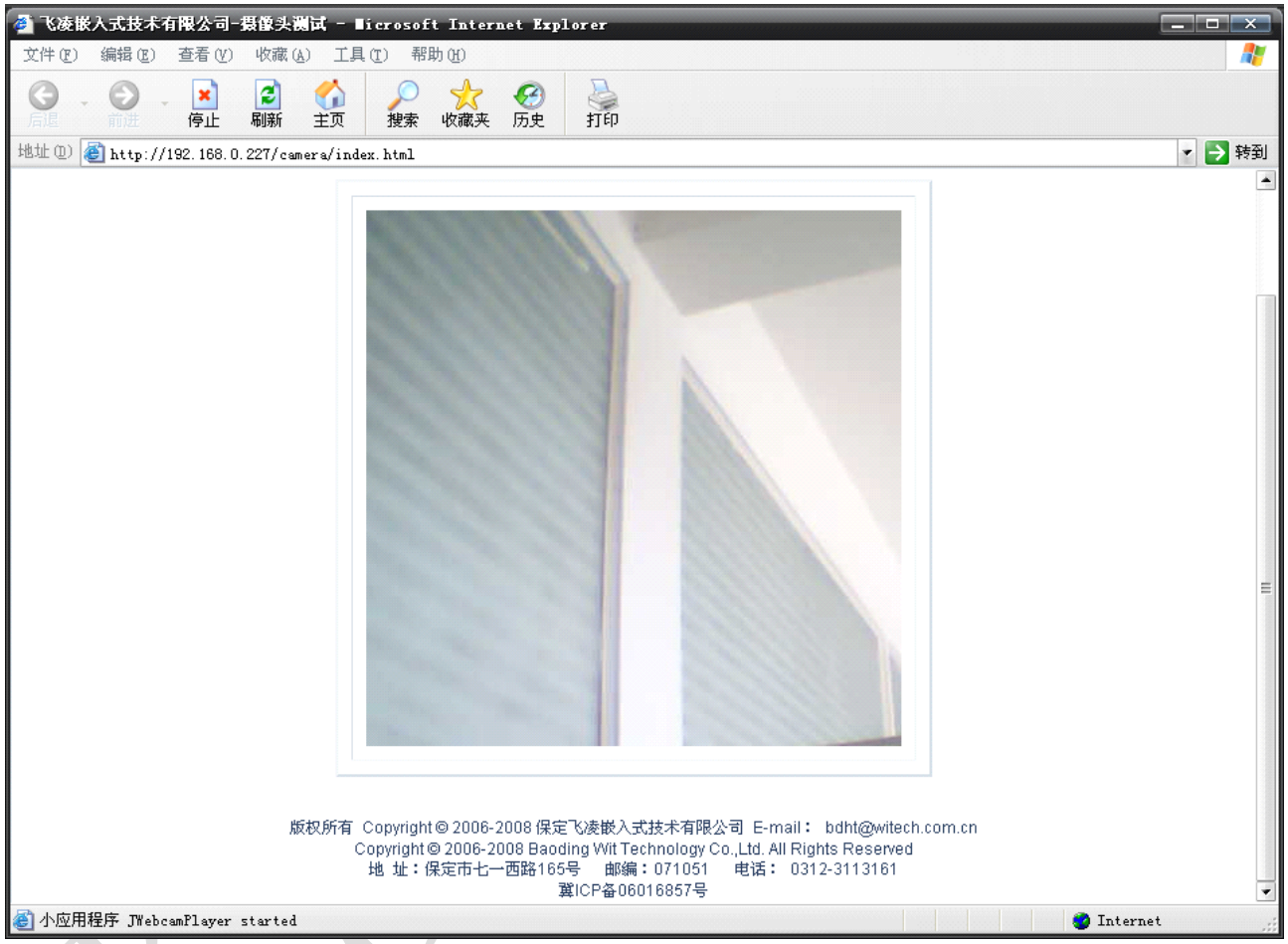


当提示 Got connection from IP 时，就说明你的 arm 板已经和你的 PC 连上了。这是你打开你的 IE 浏览器，在菜单栏选择“选项->internet 选项->连接”在连接的窗口右下角有”局域网设置”



进入局域网设置，取消代理服务器然后确定退出。在 IE 的地址栏内输入 http://192.168.0.16/camera/index.html 就可以在网页上看到摄像头相应的内容了，注意要给板子插上网线哦。

注意：摄像视频的网页显示功能还需要一个安装在 PC 机上的 JAVA 插件，名字是 j2re-1_4_2_15-windows-i586-p.exe 。



这时局域网上的其他机子输入该地址也能看到，这样就实现了远程监控。

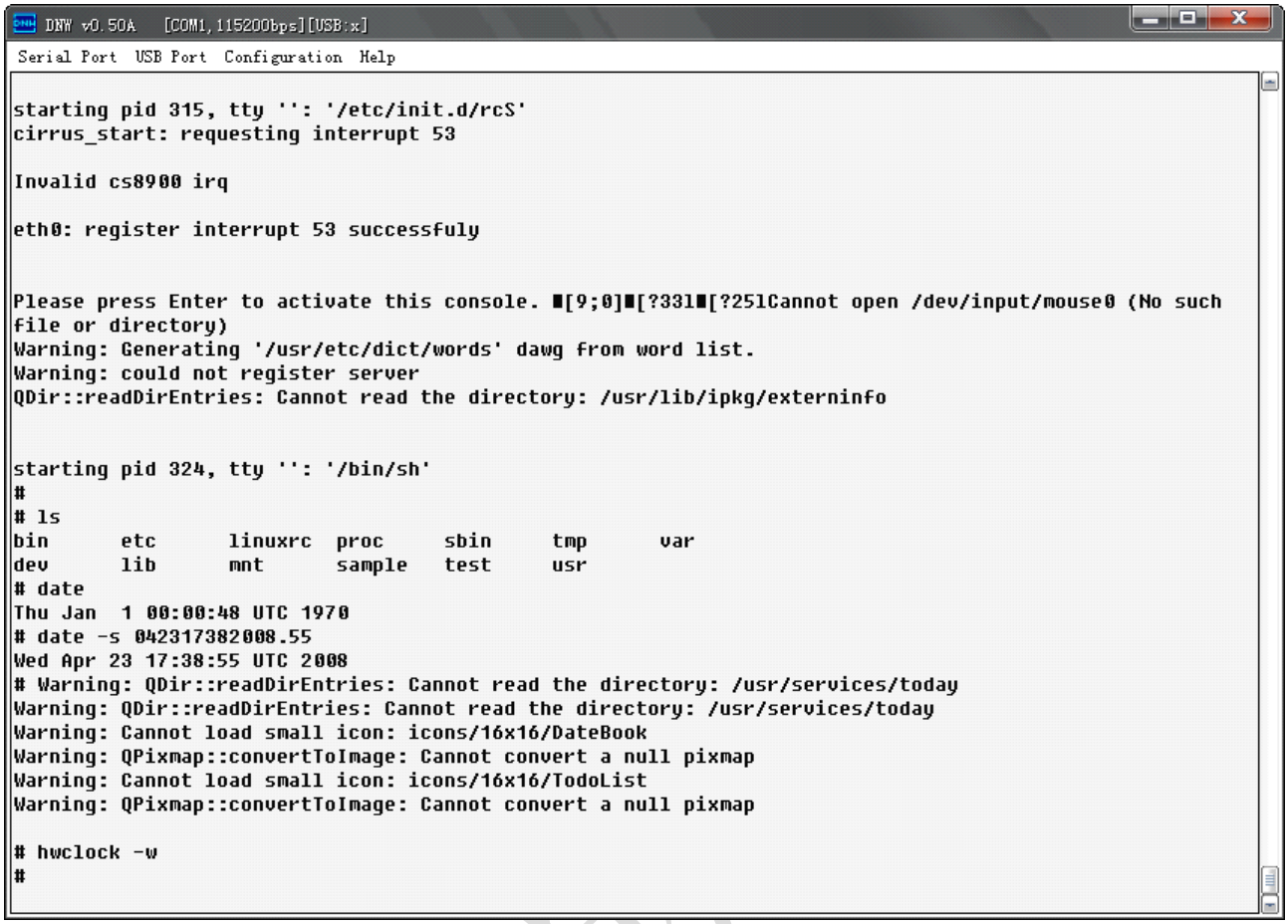
1.12 设置并保存系统时钟

Linux 中更改时间的方法一般使用 date 命令，为了把 S3C2440 内部带的时钟与 linux 系统时钟同步，一般使用 hwclock 命令进行修改。使用方法如下：

1）date –s 042317382008.55　　#该命令是将时间设置为 2008-4-23 17：38：55

2）hwclock –w　　#该命令是将刚刚设置的时间存入 S3C2440 内部的 RTC

3）若想每次上电使芯片内部时钟与 linux 系统时钟同步，则需要将命令“hwclock -s”放入/etc/init.d/rcS



1.13 测试音频 录音与播放

启动板子进入命令行后操作如下：

[1]进入 test 目录：cd /test

[2]将测试文件拷贝到/tmp 目录下：

cp record play /tmp

[3]进入/tmp 目录：# cd /tmp

[4]运行 record 开始录音：./ record（按 CTRL+C 停止）

录完后可看到/tmp 目录下多出个 audio.wav 文件即是刚才的录音文件

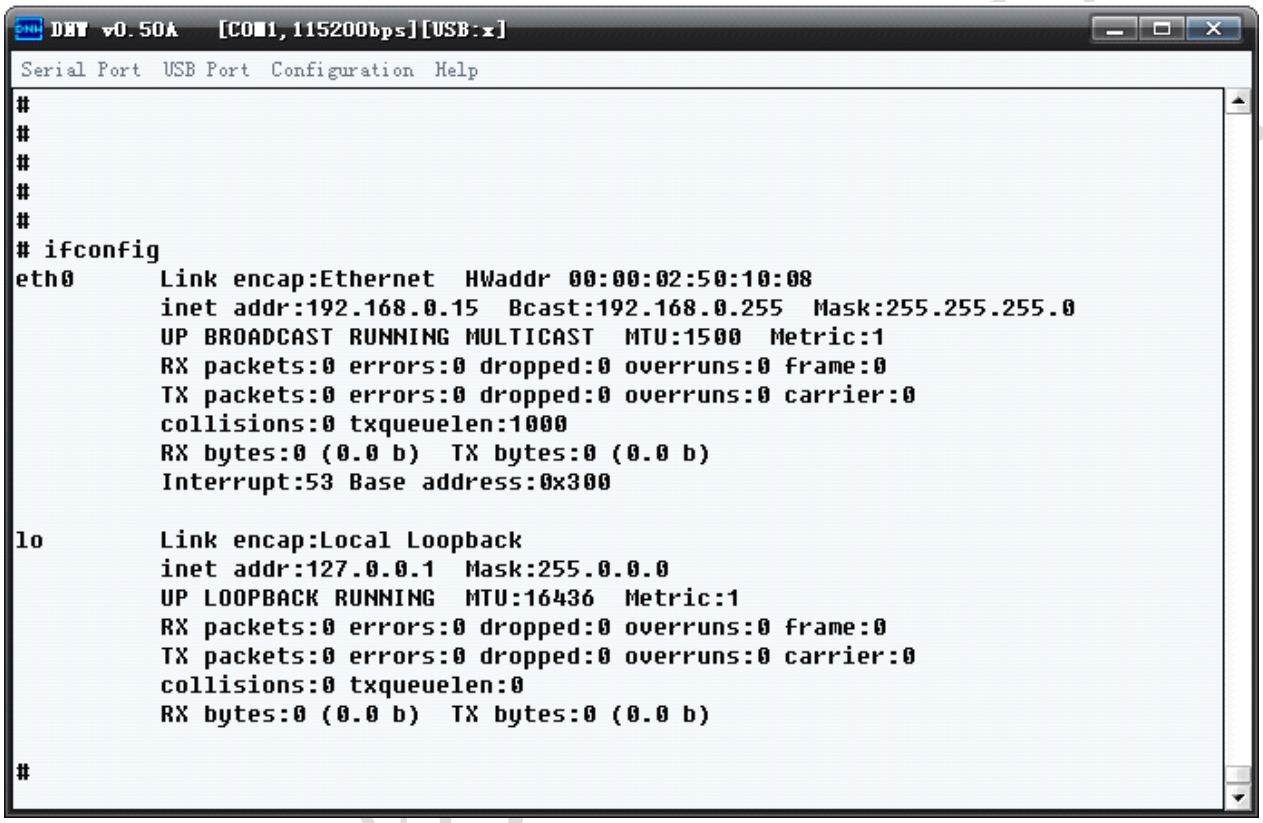
[5]运行 play 播放录音：./ play

该程序源码位于光盘目录的/linux/test/record 文件夹下。

1.14 测试网络连接

1.14.1 查看网络设置

在命令行下输入 ifconfig 可查看当前网络设置

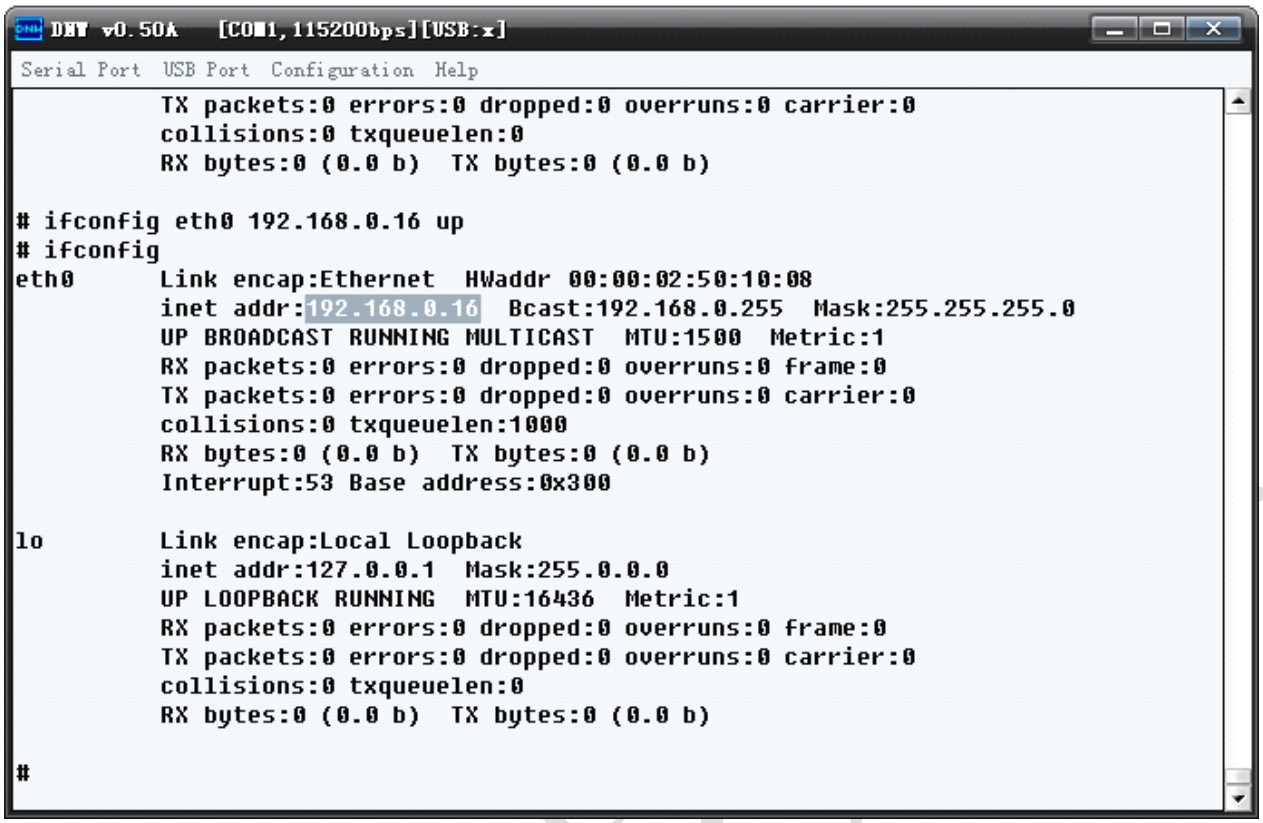


1.14.2 重新设置 IP

用下面的命令可重新设置 IP

```
ifconfig eth0 192.168.0.16 up
```

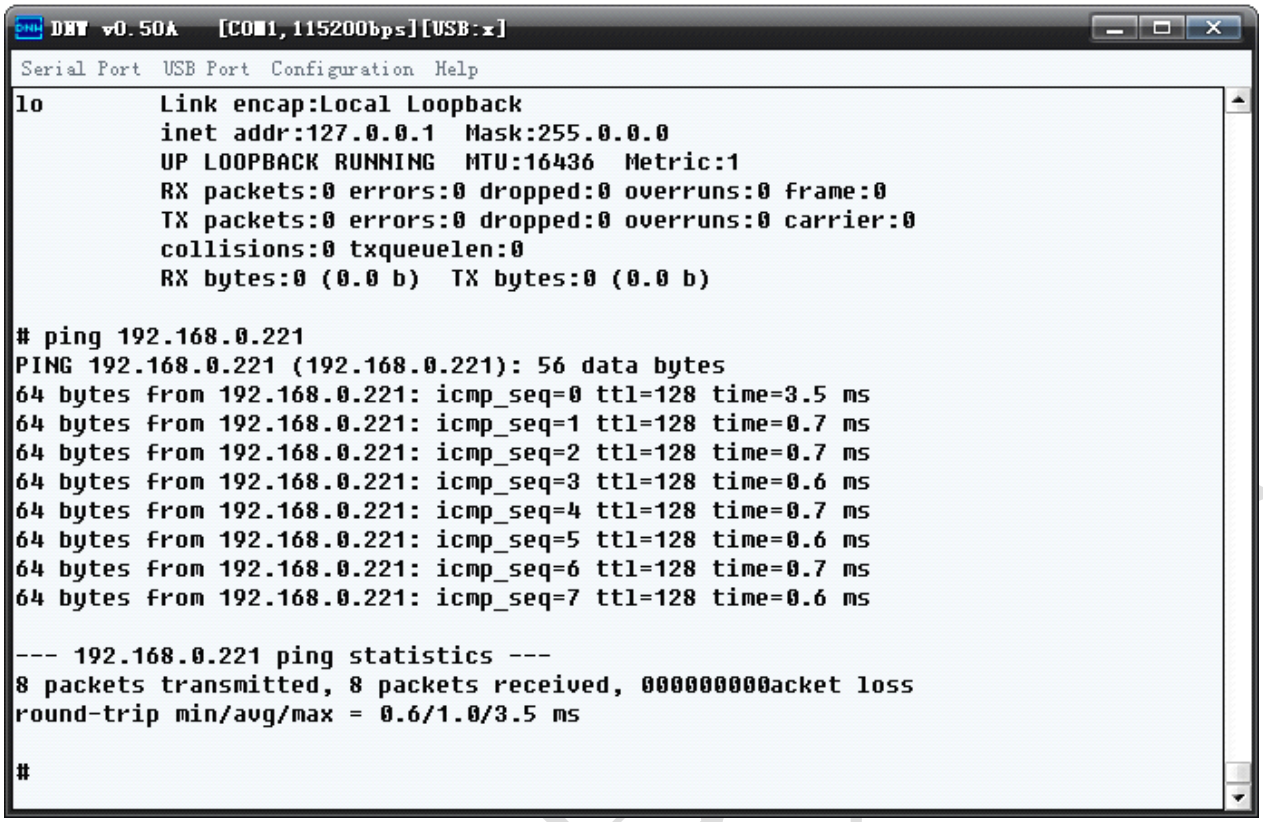
设置完成后可用 ifconfig 命令查看



1.14.3 使用 PING 命令

使用 ping 命令可检测网络是否连通，如下

ping 192.168.0.221 (通过 CTRL+C 组合键停止运行)


DNW v0.50A [COM1,115200bps][USB:x]
Serial Port USB Port Configuration Help
lo Link encap:Local Loopback
inet addr:127.0.0.1 Mask:255.0.0.0
UP LOOPBACK RUNNING MTU:16436 Metric:1
RX packets:0 errors:0 dropped:0 overruns:0 frame:0
TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:0
RX bytes:0 (0.0 b) TX bytes:0 (0.0 b)
ping 192.168.0.221
PING 192.168.0.221 (192.168.0.221): 56 data bytes
64 bytes from 192.168.0.221: icmp_seq=0 ttl=128 time=3.5 ms
64 bytes from 192.168.0.221: icmp_seq=1 ttl=128 time=0.7 ms
64 bytes from 192.168.0.221: icmp_seq=2 ttl=128 time=0.7 ms
64 bytes from 192.168.0.221: icmp_seq=3 ttl=128 time=0.6 ms
64 bytes from 192.168.0.221: icmp_seq=4 ttl=128 time=0.7 ms
64 bytes from 192.168.0.221: icmp_seq=5 ttl=128 time=0.6 ms
64 bytes from 192.168.0.221: icmp_seq=6 ttl=128 time=0.7 ms
64 bytes from 192.168.0.221: icmp_seq=7 ttl=128 time=0.6 ms
--- 192.168.0.221 ping statistics ---
8 packets transmitted, 8 packets received, 000000000acket loss
round-trip min/avg/max = 0.6/1.0/3.5 ms
#

上面的信息说明网络连接正常。

1.14.4 如何 DHCP 动态获取 IP 和 DNS

【1】在 PC 机上配置 DHCP 服务

确定你的 PC 端 linux 系统安装了 dhcp 服务程序，如何没有安装 REDHAT 光盘中有安装文件 dhcp-3.0pl1-23.i386.rpm

#rpm –ivh dhcp-3.0pl1-23.i386.rpm

安装好后在 etc 目录下写一个 dhcpd.conf 文件，用来保存服务器配置信息，程序执行时会读取这个文件的配置信息，这个文件的内容：

```
subnet 192.168.0.0 netmask 255.255.255.0{
    range 192.168.0.10    192.168.0.30;
    default-lease-time 86400;
    max-lease-time 604800;
    option subnet-mask 255.255.255.0;
```

option routers 192.168.0.2;

option domain-name "cy.com";

option broadcast-address 192.168.0.255;

option domain-name-servers 192.168.0.3; }

subnet x.x.x.x netmask x.x.x.x 说明 IP 地址是否属于该子网上，提供子网的一些参数；

range x.x.x.x x.x.x.x 是 DHCP 服务器可以分配的 IP 地址范围；

default-lease-time 设置缺省的 IP 租用时间，常用的是 86400 秒(一天)；max-lease-time 最大租用时间 。常用的是 604800(一周)；

option subnet-mask 设置 IP 地址的子网掩码；option routers 设置在 DHCP 发布 IP 地址的同时，把网关发布出去，这一项是用来指明网关的；

option domain-name-servers 设置在 DHCP 发布 IP 地址的同时,把 DNS 发布出去,这一项用来指明 DNS 服务器的 IP 地址；

option broadcast-address 设置该子网的广播地址；option doamin-name 设置 DNS 域名。

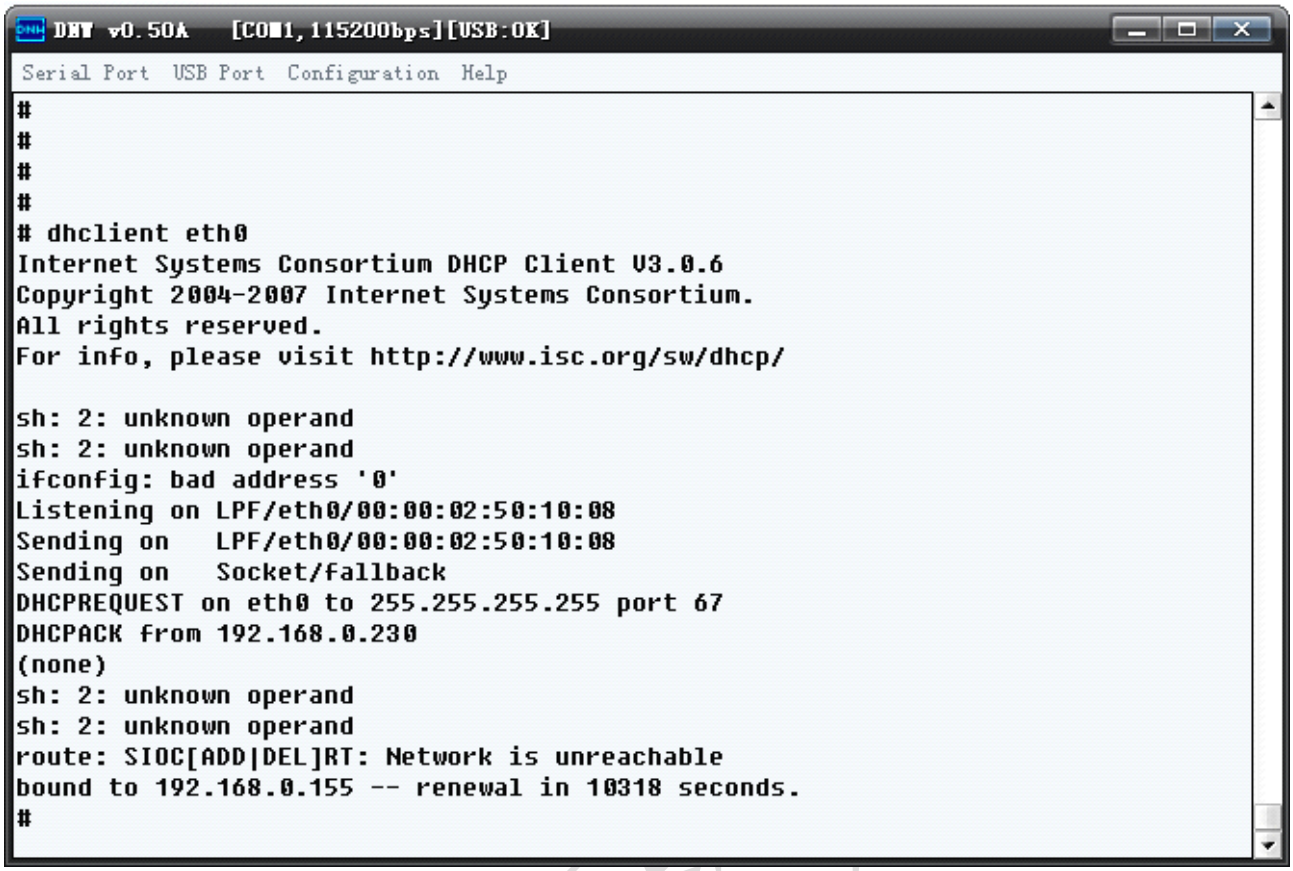
完成上面的配置信息后，就可以启动 dhcp 服务了

#service dhcpd start

【2】在板子上执行客户端程序

板子上电启动 linux 操作系统并进入命令行，在命令行上输入如下命令：

dhclient eth0 (会显示如下内容)



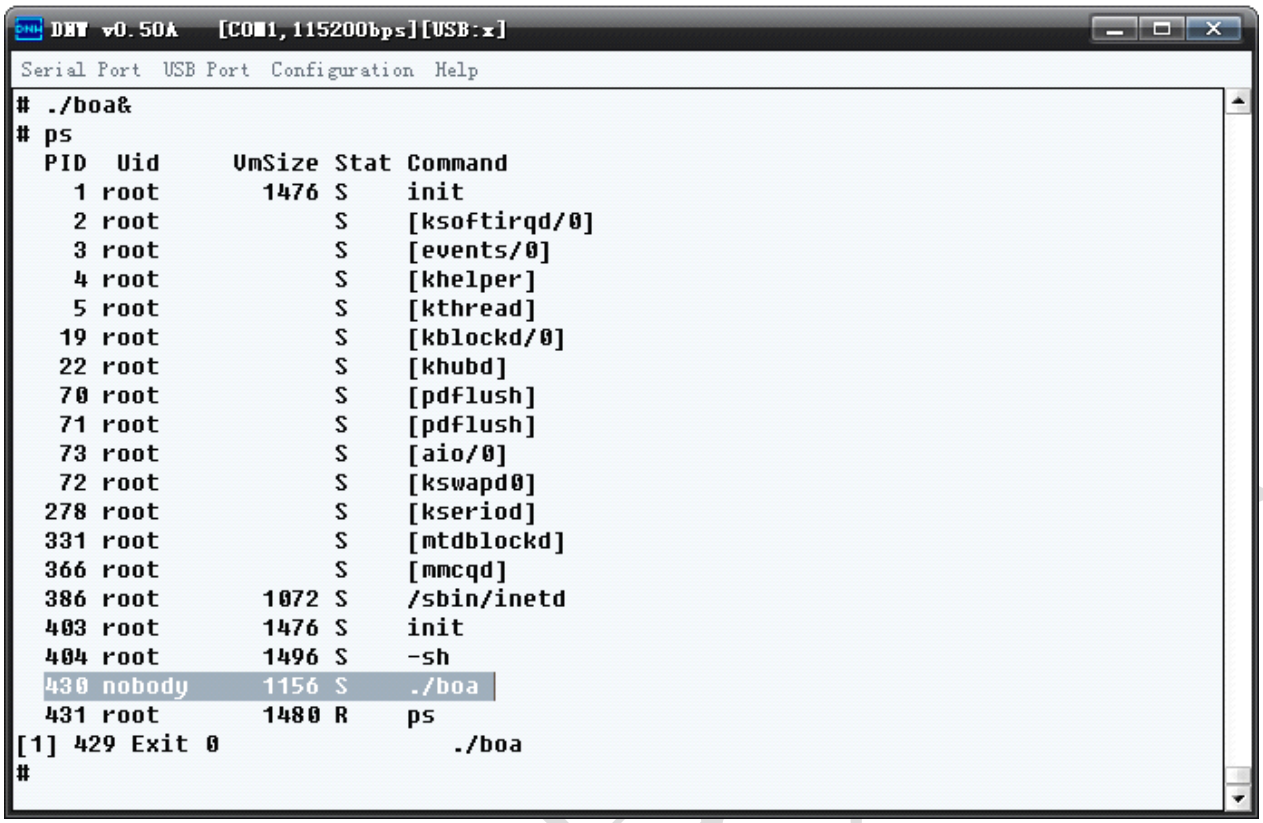
分配到的 IP 为 192.168.0.155，可通过 ifconfig 命令查看。

1.15 WebServer 功能

1.15.1 启动 WebServer

进入 Linux 命令行输入如下命令：

boa&　　(&符号表示在后台运行)



这时开发板就已经作为一个 web 服务器在运行了。

BOA Webserver 是一个小巧高效的 web 服务器，是一个运行于 unix 或 linux 下的支持 CGI 的、非常合适于嵌入式系统的单任务的 http 服务器，源代码开放、性能高。

1.15.2 浏览 Web 服务器上的网页

打开 IE 在地址栏里输入 http://192.168.0.15/index.html

1.16 使用 telnet 远程登录开发板

1. 关于 telnet

Telnet 是传输控制协议 / 因特网协议（TCP / IP）网络（例如 Internet）的登录和仿真程序。它最初是由 ARPANET 开发的，但是现在它主要用于 Internet 会话。它的基本功能是，允许用户登录进入远程主机系统。

在 telnet 中你可以做一些相关工作如：查看系统信息、传递文件、清除日志等等，有点像黑客。

telnet 成功的条件：

A. 在本地计算机上必须装有包含 telnet 协议的客户程序

B. 必须知道远程主机的 IP 地址或域名，并开启了 telnet 服务

C. 必须知道登录标识与口令

2. 如何 telnet 远程登录开发板

开发板上电正常运行后，在控制台输入 telnetd 命令启动 telnet 服务，此时当你用 ps 命令查看进程时会有叫 telnetd 的进程在运行，说明 telnet 服务已经启动，这时候用户就可以通过网络来远程登录开发板了。

在 windows 的命令窗口下输入 telnet 192.168.0.15,片刻后就会出来如图所示的登录界面

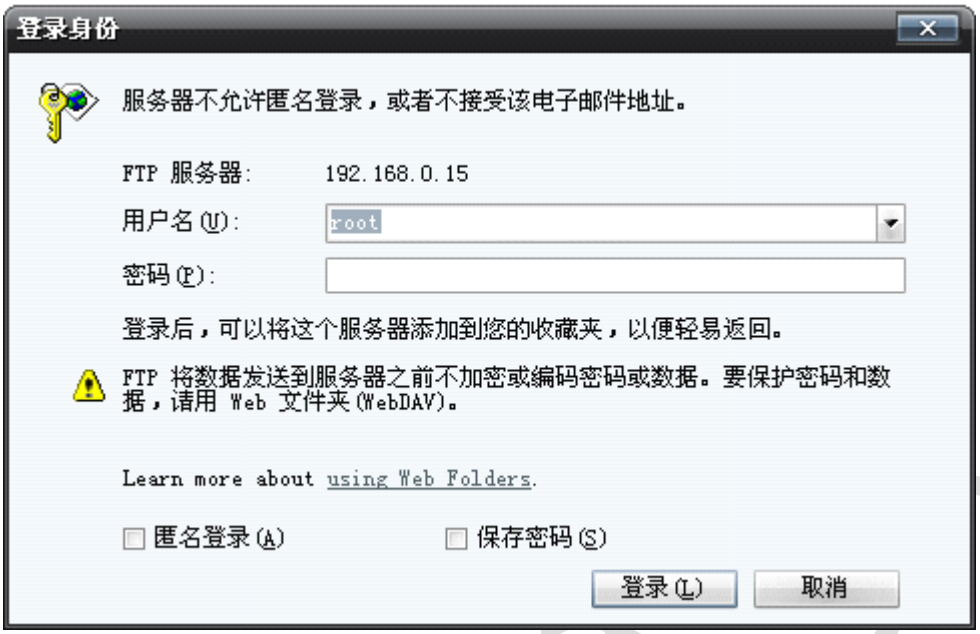
（开始->运行，输入 cmd 回车，即可进入 windows 的命令窗口）

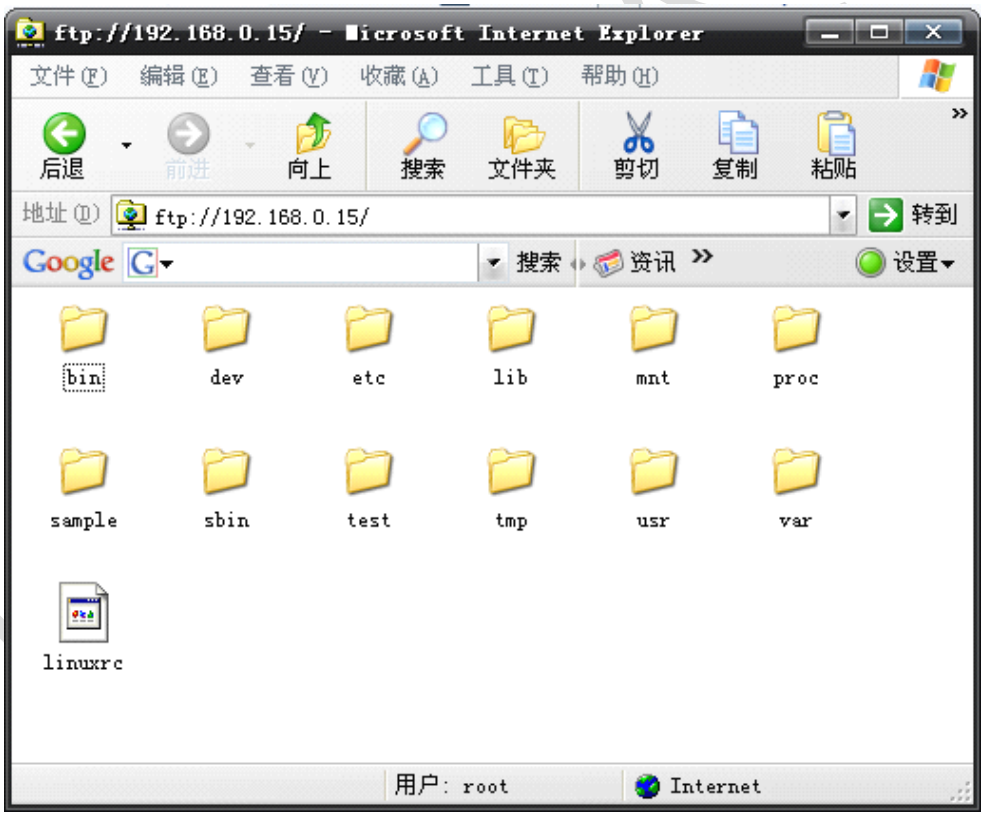
输入“root”并回车（没有密码），如果成功就会像下图一样进入命令行，可以对开发板做相关操作。

1.17 使用 ftp 传递文件

无论在 linux 系统还是在 windows 系统中，一般都自带一个命令行的 ftp 命令程序，使用 ftp 可以登陆远程的主机，并传递文件，当然这需要主机提供 ftp 服务和相应的权限。

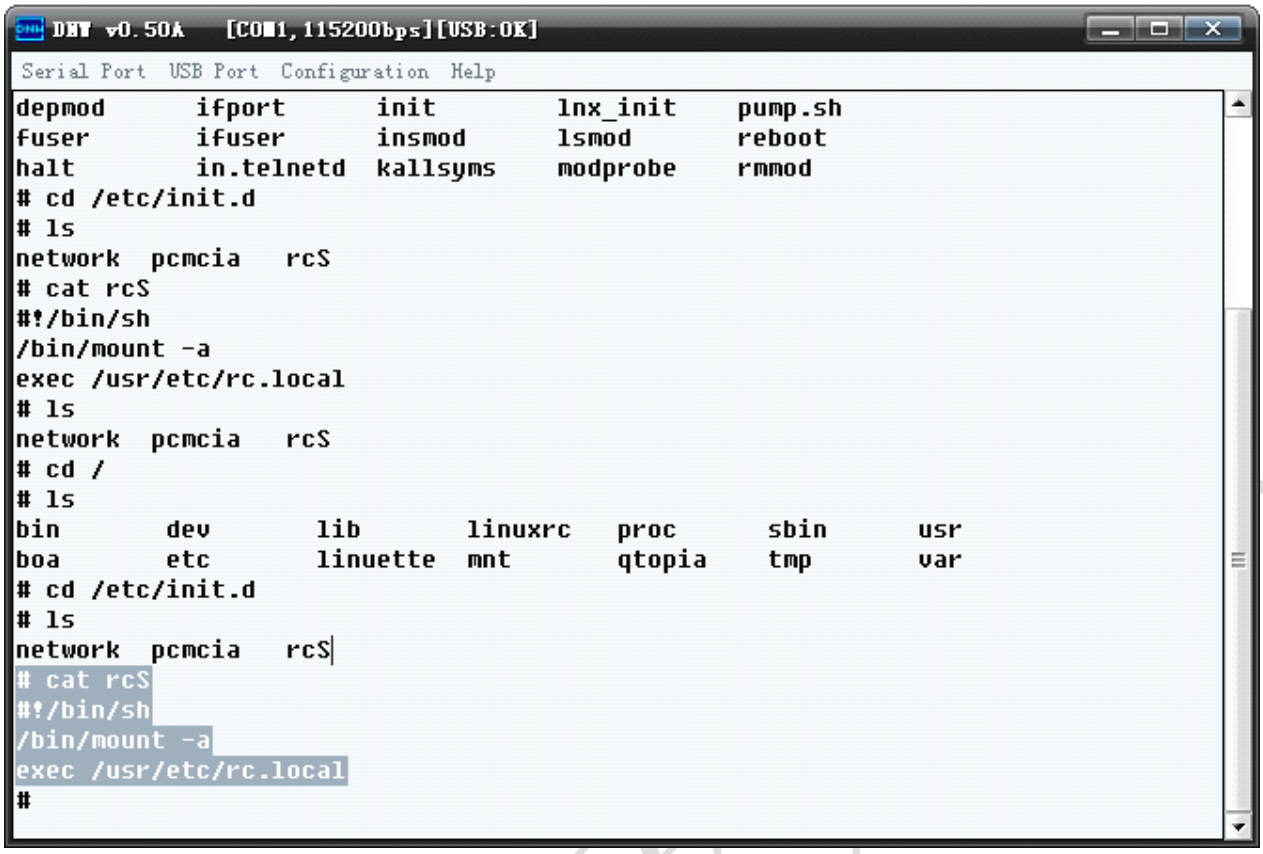
本开发板提供了 ftp 命令，首先启动 ftp 服务：在控制台输入 vsftpd& 。启动服务后就可以用 ftp 软件登录了，用户名为 root，密码为空（也可用 IE 登录）。





1.18 如何开机自动运行程序

Linux 的启动脚本位于/etc/init.d/rcS



若想自启动自己的应用程序可在后面添加，然后重做文件系统并烧写（制作文件系统参见 linux 教程 ）。如开机自动运行 Web 服务，可在后面添加如下代码：

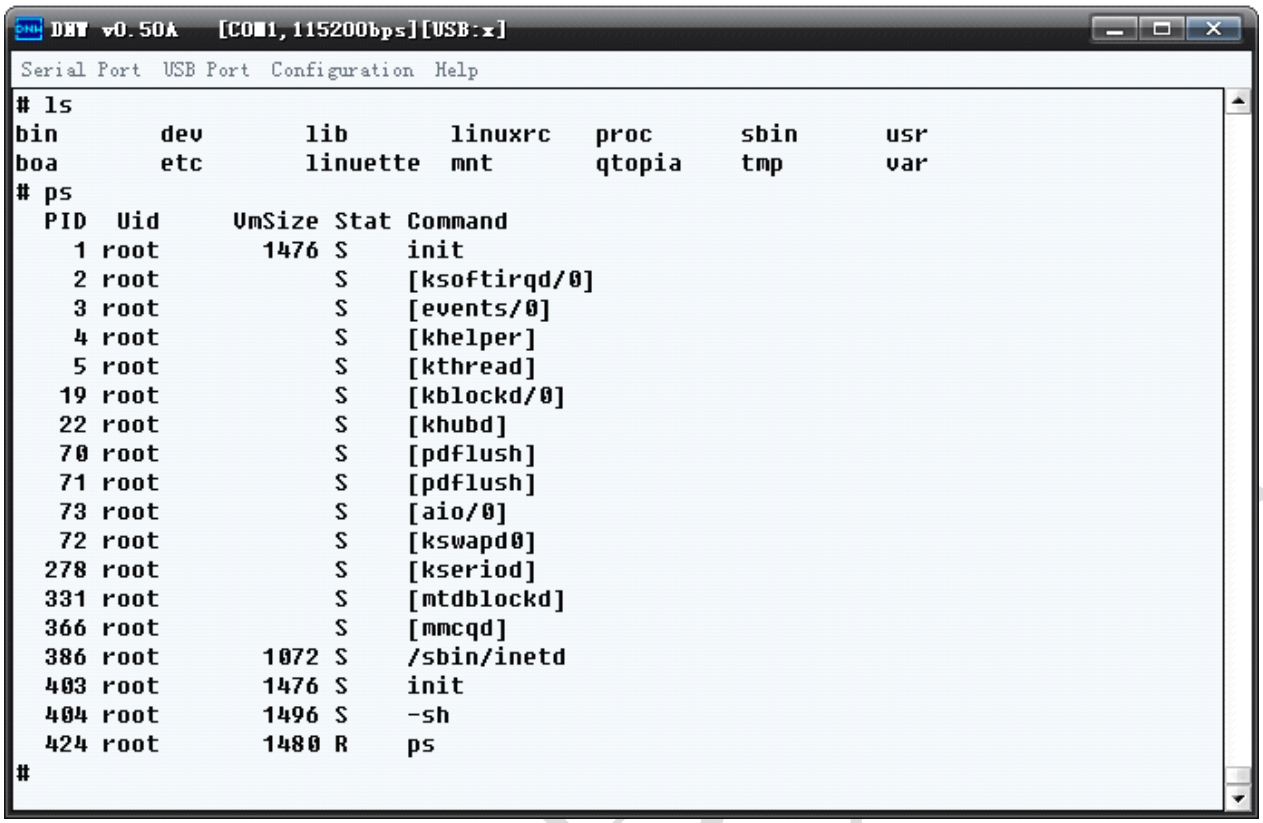
```
boa&
```

1.19 如何中止程序运行

使用 Ctrl+C 的组合键，可以中止正在运行的程序，如果是在后台运行的程序，需要使用 kill 来杀掉对应的进程即：

```
kill 进程号
```

用 ps 命令可查看当前正在运行的进程。


DNW v0.50A [COM1,115200bps][USB:x]
Serial Port USB Port Configuration Help
ls
bin dev lib linuxrc proc sbin usr
boa etc linuette mnt qtopia tmp var
ps
PID Uid VmSize Stat Command
1 root 1476 S init
2 root S [ksoftirqd/0]
3 root S [events/0]
4 root S [khelper]
5 root S [kthread]
19 root S [kblockd/0]
22 root S [khubd]
70 root S [pdflush]
71 root S [pdflush]
73 root S [aio/0]
72 root S [kswapd0]
278 root S [kseriod]
331 root S [mtdblockd]
366 root S [mmcqd]
386 root 1072 S /sbin/inetd
403 root 1476 S init
404 root 1496 S -sh
424 root 1480 R ps
#

二.Linux 开发教程

本篇中要求开发环境为 Linux 操作系统，建议使用 Redhat9.0

Linux 系统的主要特点

设备独立性：是指操作系统把所有外部设备统一当作成文件来看待，只要安装它们的驱动程序，任何用户都可以象使用文件一样，操纵、使用这些设备，而不必知道它们的具体存在形式。Linux 是具有设备独立性的操作系统，它的内核具有高度适应能力

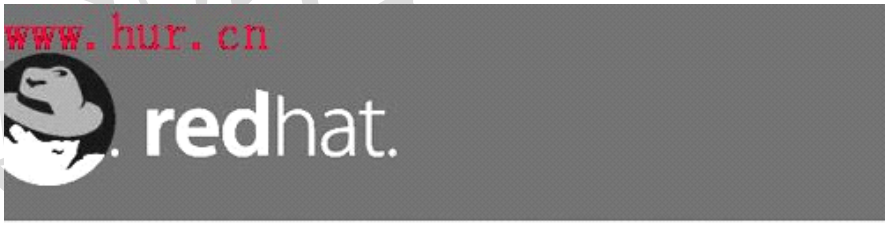
提供了丰富的网络功能：完善的内置网络是 Linux 一大特点。

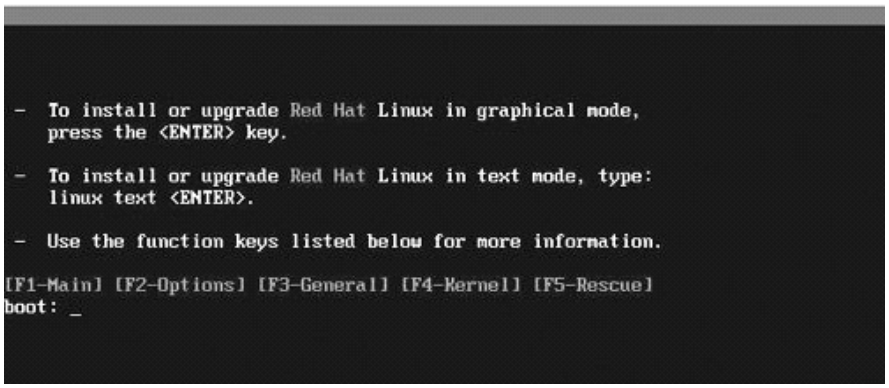
可靠的安全系统：Linux 采取了许多安全技术措施，包括对读、写控制、带保护的子系统、审计跟踪、核心授权等，这为网络多用户环境中的用户提供了必要的安全保障。

良好的可移植性：是指将操作系统从一个平台转移到另一个平台使它仍然能按其自身的方式运行的能力。Linux 是一种可移植的操作系统，能够在从微型计算机到大型计算机的任何环境中和任何平台上运行。

2.1 如何安装 RedHat 9.0

最简单，最方便的安装方法当然是从 CD 安装，你可以享受最人性化的，类似于 Windows 的安装。你只要将计算机设置成光驱引导，把安装 CD1 放入光驱，重新引导系统，在安装界面中直接按回车，即进入如下图形化安装界面：





由图可见，在提供“豪华”的图形化 GUI 安装界面的同时，REDHAT Linux 9.0 仍然保留了以往版本中的字符模式安装界面，这对于追求安装速度与效率的用户一直是很有吸引力的。因为许多用户是将 RedHat 9 安装成服务器来使用的，不需要 X-Window 以及 GUI 安装界面。

RedHat 9 的安装步骤中比以往多了一个环节，那就是对安装光盘介质的检测。它允许在开始安装过程前对安装光盘介质进行内容校验，以防止在安装的中途由于光盘无法读取或是内容错误造成意外的安装中断，导致前功尽弃。

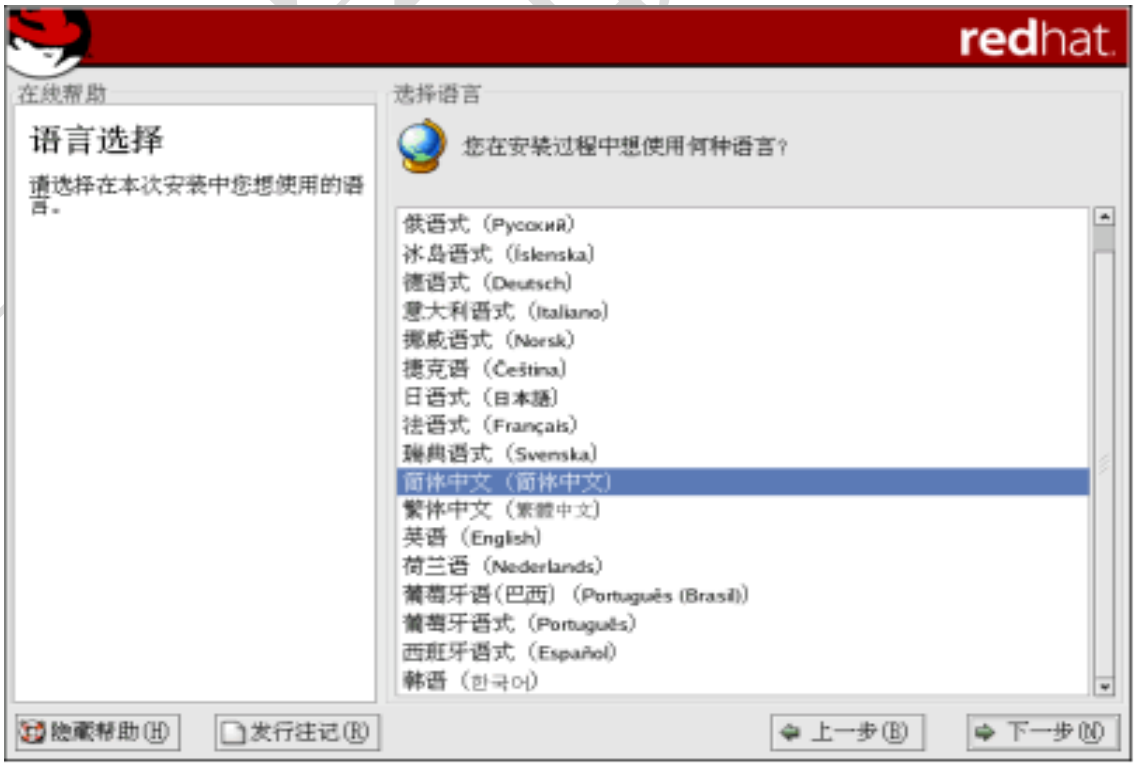
REDHAT Linux 9.0 如果完全安装将达到 7 张光盘，安装时间长达几十分钟。如果因为一张光盘的内容错误导致安装失败，这将浪费很多安装时间。所以，建议在安装之前对光盘进行介质检测与校验以保证安装顺利进行。

当然也可以通过硬盘安装，但相对比较麻烦，再次不再讲述，有兴趣的朋友可以到网上查些资料。

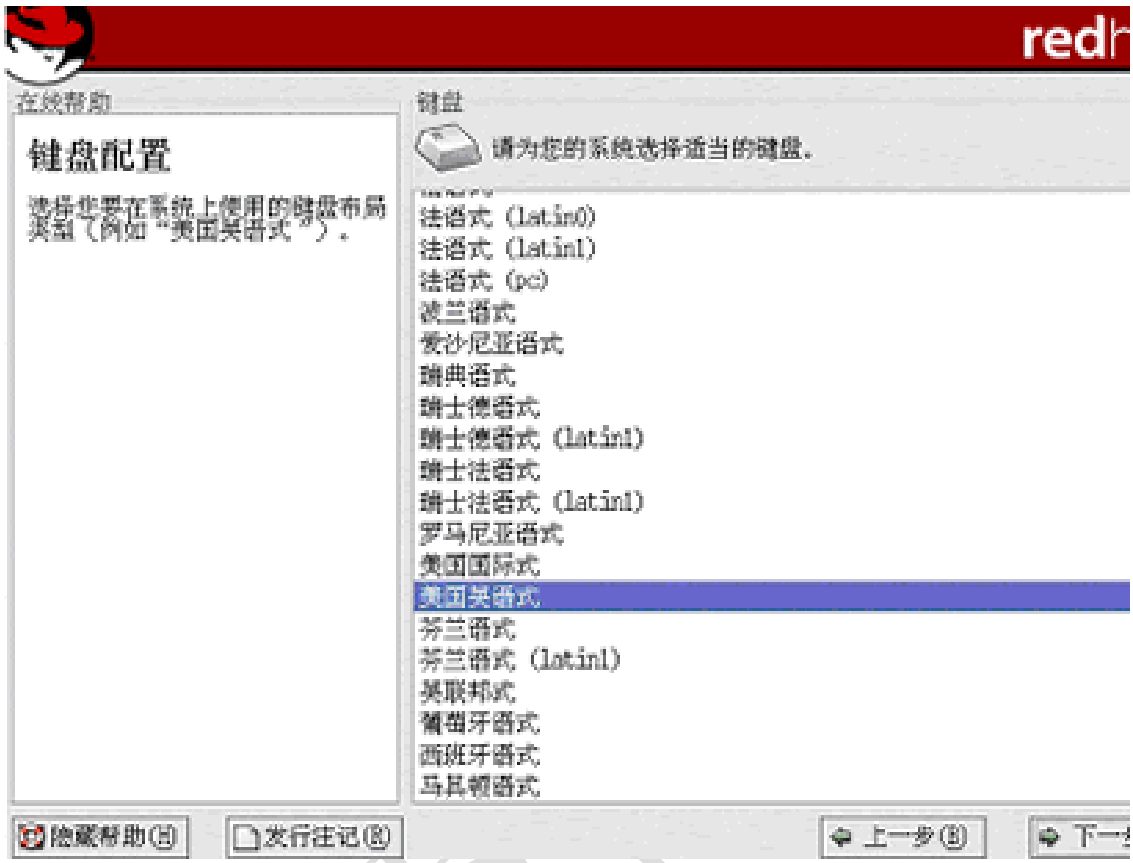
接下来就可以按提示安装了

1） 语言选择：

RedHat 支持世界上几乎所有国家的语言，这里只要选中简体中文，并将系统默认语言选择为简体中文如图，那么在安装过程结束，系统启动后，整个操作系统的界面都将是简体中文的了，用户不用做任何额外的中文化操作和设置



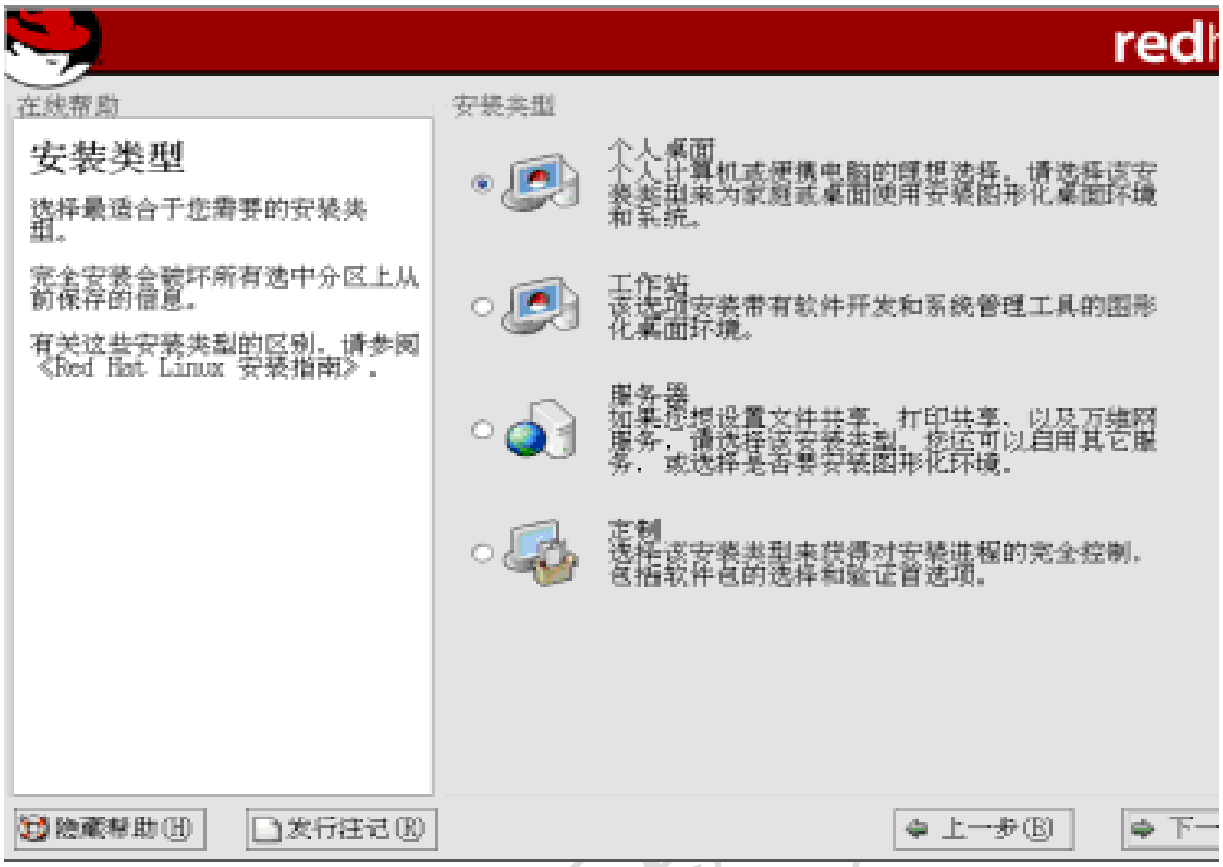
2）键盘配置：


在线帮助
键盘配置
请为您的系统选择适当的键盘。
键盘
请为您要在系统上使用的键盘布局类型（例如“美国英语式”）。
法语式 (latin0)
法语式 (latin1)
法语式 (pc)
波兰语式
爱沙尼亚语式
瑞典语式
瑞士德语式
瑞士德语式 (latin1)
瑞士法语式
瑞士法语式 (latin1)
罗马尼亚语式
美国国际式
美国英语式
芬兰语式
芬兰语式 (latin1)
英联邦式
葡萄牙语式
西班牙语式
马其顿语式
隐藏帮助(H)
发行注记(R)
上一步(B)

3）鼠标配置：

4）选择安装还是升级：

5） 安装类型：



6）磁盘分区设置：

这也许是整个安装过程中惟一需要用户较多干预的步骤，REDHAT Linux 9.0 提供了两种分区方式——自动分区和使用 DISK DRUID 程序进行手动分区

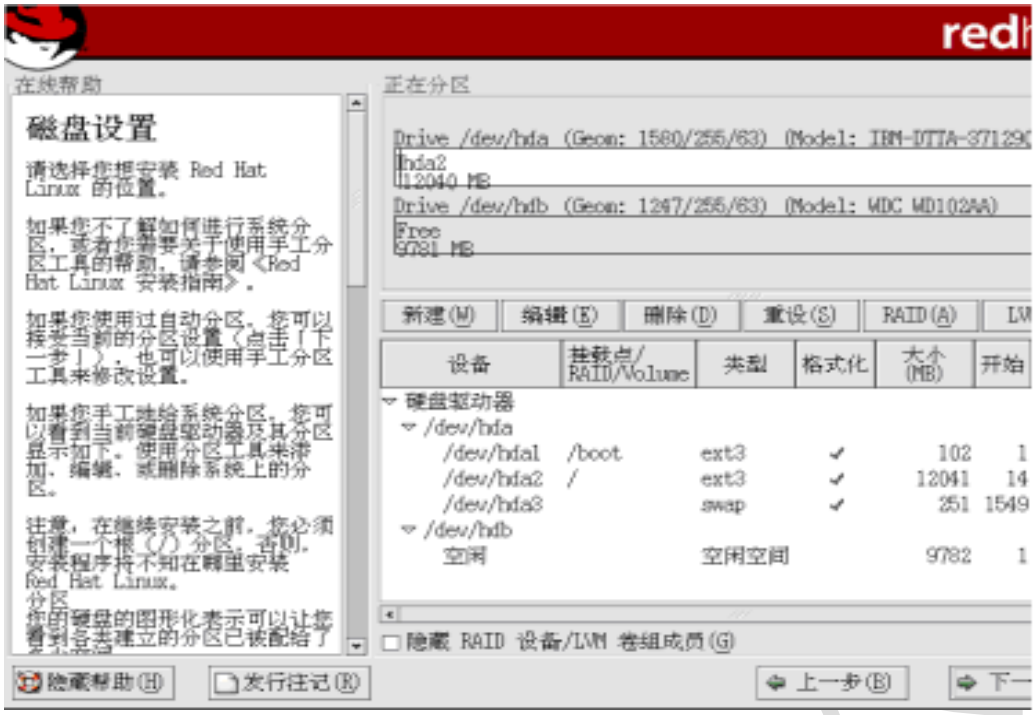
（1）自动分区：如果是全新的计算机，上面没有任何操作系统，建议使用“自动分区”功能，它会自动根据磁盘以及内存的大小，分配磁盘空间和 SWAP 空间。

这是一个“危险”的功能，因为它会自动删除原先硬盘上的数据并格式化成为 Linux 的分区文件系统（EXT3、REISERFS 等），所以除非计算机上没有任何其他操作系统或是没有任何需要保留的数据，你才可以使用“自动分区”功能。

（2）手动分区：如果硬盘上有其他操作系统或是需要保留其他分区上的数据，建议采用 DISK DRUID 程序进行手动分区。DISK DRUID 是一个 GUI 的分区程序，它可以对磁盘的分区进行方便的删除、添加和修改属性等操作，它比以前版本中使用的字符界面 Fdisk 程序的界面更加友好，操作更加直观。下面我们来看看如何使用 DISK DRUID 程序对硬盘进行分区。

因为 Linux 操作系统需要有自己的文件系统分区，而且 Linux 的分区和微软 Windows 的分区不同，不能共用，所以，需要为 Linux 单独开辟一个（或若干个）分区。Linux 一般可以采用 EXT3 分区，这也是 REDHAT Linux 9.0 默认采用的文件系统。

7）为系统分区：



为 Linux 建立文件分区可以有两种办法，一种是利用空闲的磁盘空间新建一个 Linux 分区，另一种是编辑一个现有的分区，使它成为 Linux 分区。如果没有空闲的磁盘空间，就需要将现有的分区删除后，腾出空间，以建立 Linux 分区。

DISK DRUID 程序中有明显的新建、删除、编辑、重设等按钮。用户可以直观地对磁盘进行操作。在使用 DISK DRUID 对磁盘分区进行操作时，有四个重要的参数需要仔细设定：它们是挂载点、文件系统类型、驱动器、分区大小

挂载点：它指定了该分区对应 Linux 文件系统的哪个目录，Linux 允许将不同的物理磁盘上的分区映射到不同的目录，这样可以实现将不同的服务程序放在不同的物理磁盘上，当其中一个物理磁盘损坏时不会影响到其他物理磁盘上的数据。

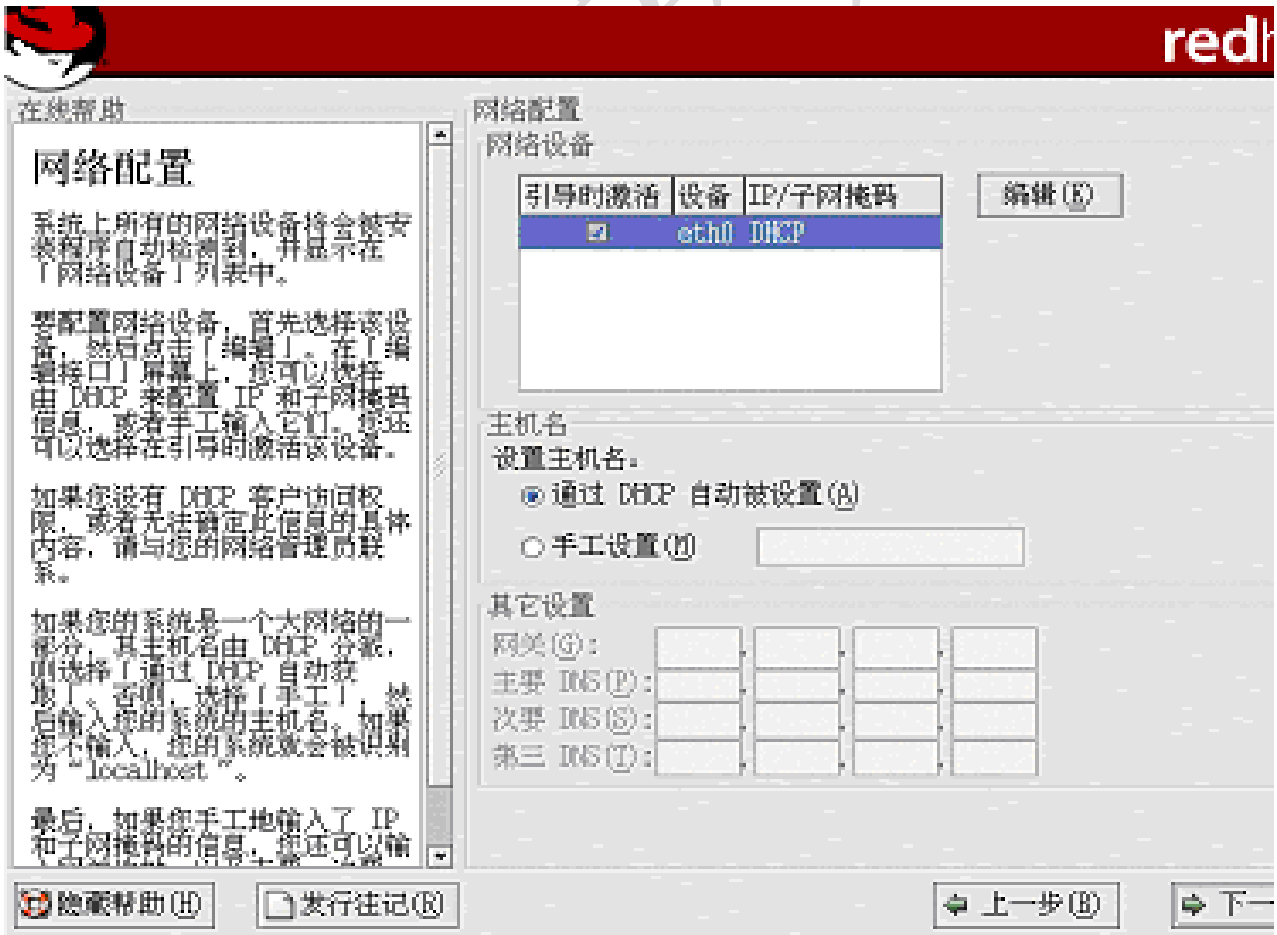
文件系统类型：它指定了该分区的文件系统类型，可选项有 EXT2、EXT3、REISERFS、JFS、SWAP 等。Linux 的数据分区创建完毕后，有必要创建一个 SWAP 分区，它实际上是用硬盘模拟的虚拟内存，当系统内存使用率比较高的时候，内核会自动使用 SWAP 分区来模拟内存。

大小：指分区的大小（以 MB 为单位），Linux 数据分区的大小可以根据用户的实际情况进行填写，而 SWAP 大小根据经验可以设为物理内存的两倍，但是当物理内存大于 1GB 时，SWAP 分区可以设置为 2GB。

允许的驱动器：如果计算机上有多个物理磁盘，就可以在这个菜单选项中选中需要进行分区操作的物理磁盘。

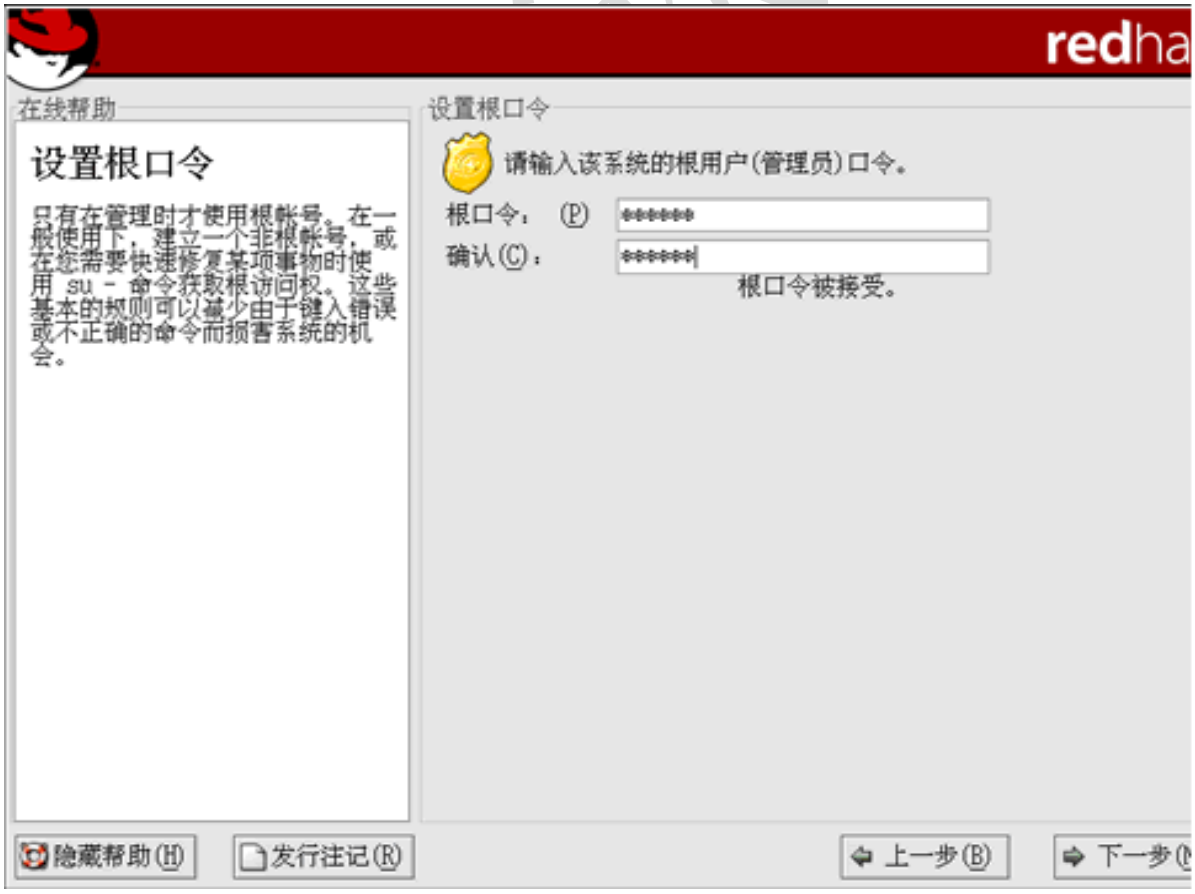
经过磁盘分区的操作，安装过程中相对最复杂的一个步骤已经过去，接下来的安装将是一马平川。

8) 网络配置：



9) 防火墙配置：

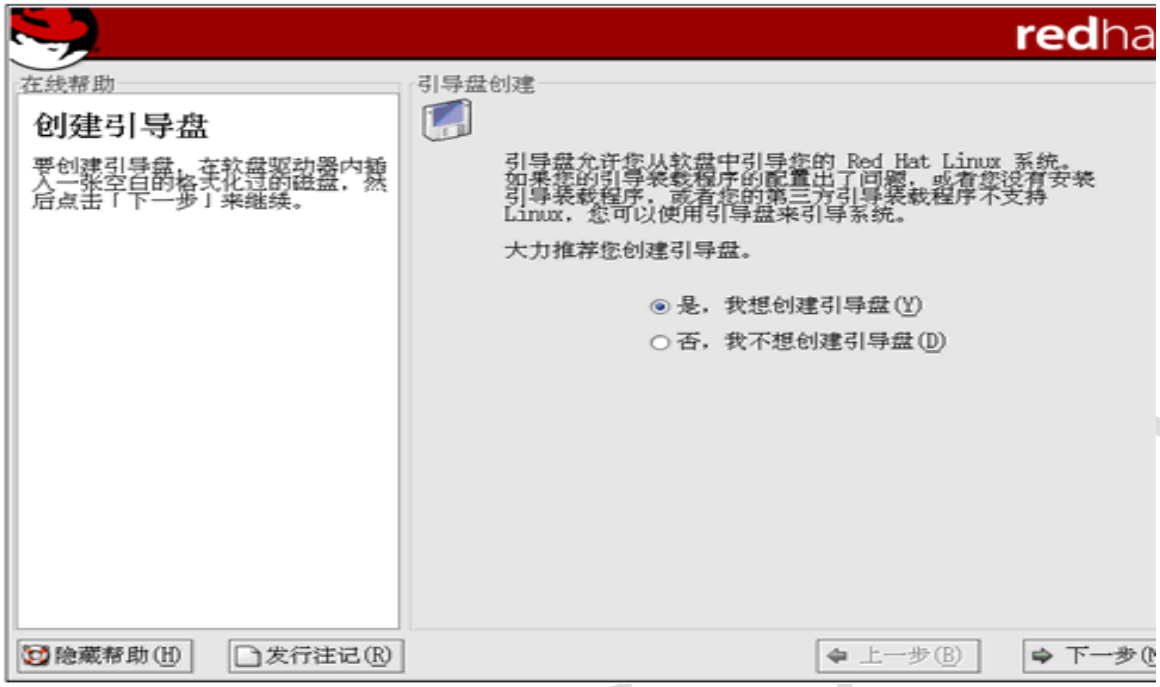
10) 设置根口令



11) 选择软件包组： （注：这步要小心）

12）安装软件包： （这步需要一定的时间）

13）创建引导盘：（是与否问题不大）



这样你的系统就顺利装好了。

使用 Red Hat Linux 系统的一个前提条件是登录。登录实际上是向系统做自我介绍，又称验证（authentication）。如果键入了错误的用户名或口令，就不会被允许进入系统。

2.2 在 RedHat 中添加新用户

Linux 是一个多用户和多任务的操作系统。用户要登录 Linux 首先要有个合法的账号，所以必须创建平常使用的用户账号。

现在我们以 root(管理员账号)身份登录 RH9 系统进行账户管理为例说明：在主菜单中选择“系统设置→用户和组群”，弹出“Redhat 用户管理器”对话框，点击工具栏的“添加用户”按钮，弹出“创建新用户”对话框如下图，在“用户名”栏中输入用户账号名，然后在“口令”和“确认口令”栏中输入账号密码，其他的设置可以使用系统的默认设置，一般不需要手工更改。最后点击“确定”按钮。

注意：要有管理员权限才能创建新账号。

除了以上方式，还可以在终端命令行窗口中创建用户账号：点击 GNOME 面板中的终端命令行窗口快捷工具按钮，在“#”提示符下输入“adduser tjrao1027 -p 123wefg”命令，其中“tjrao1027”为账号名，“123wefg”为账号密码。

小提示：为什么要创建新用户

在 Linux 中，如果你以 root 身份登录系统，则可以对系统进行任何更改，包括对内核的重新编译，这样很容易造成对系统的损坏。所以在一般情况下，我们要创建另外的用户来进行日常使用，root 用户只在系统维护、故障排除等情况下使用。

2.3 建立交叉编译环境

什么是交叉编译呢？

在一种计算机环境中运行的编译程序，能编译出在另外一种环境下运行的代码，我们就称这种编译器支持交叉编译。这个编译过程就叫交叉编译。简单地说，就是在一个平台上生成另一个平台上的可执行代码。如 keil 软件，在 keil 上编译，但在单片机上运行，典型的交叉编译。

编译嵌入式 linux 内核前，要先安装交叉编译工具 toolchain，随板光盘已附带编译工具。

编译工具 cross-3.4.1.tar.bz2，安装此编译器只需要在/usr/local 目录下建一个 arm 的目录 ，先将光盘目录“OK2440\实用工具\linux 交叉编译工具链”文件夹下的 cross-3.4.1.tar.bz2 拷贝到某个目录下，然后进入该目录，执行解包命令 tar xjvf cross-3.4.1.tar.bz2 即可，之后可编辑/etc/bashrc 文件，在最后增加路径 export PATH=/usr/local/arm/3.4.1/bin:$PATH，以后编译内核或其他应用程序均可用 arm-linux- 来指定交叉编译器。可以使用 echo $PATH 来查看环境变量中是否有 /usr/local/arm/3.4.1/bin 路径变量。

2.4 编译内核

1、解压内核包

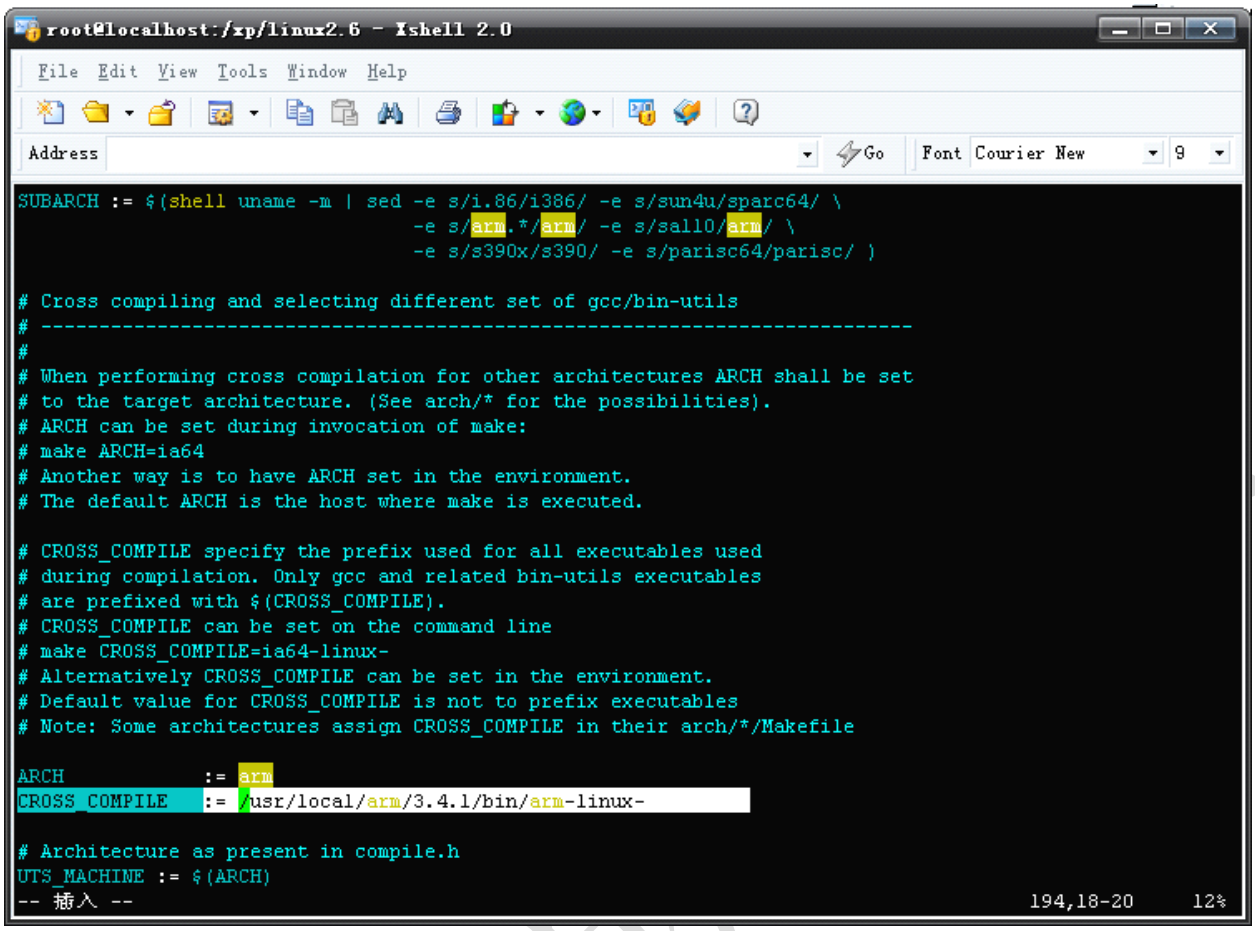
先将光盘目录的“OK2440\linux”的 Linux 内核包拷贝到某个目录下，进入这个目录，然后解压这个 Linux 源码包，命令：

```
tar -jxvf   OK2440_kernel2.6.12.tar.bz2
```

2、编辑 Makefile 文件

进入解压的目录后，运行命令： vi Makefile 找到 “CROSS_COMPILE=”这行，将它改为 CROSS_COMPILE=/usr/local/arm/3.4.1/bin/arm-linux-（或者你的系统所安装的 ARM-Linux-gcc 的编译器路径）。设置好后保存退出。这一行是指明所用交叉编译器的版本，位置。

修改“ARCH :=”为“ARCH :=arm”指定处理器。



3、 配置内核

make config（基于文本的最为传统的配置界面，不推荐使用）

make menuconfig（基于文本选单的配置界面，字符终端下推荐使用）

make xconfig（基于图形窗口模式的配置界面，Xwindow 下推荐使用）

make oldconfig（如果只想在原来内核配置的基础上修改一些小地方，会省去不少麻烦）

这三个命令中，make xconfig 的界面最为友好，但如果你不能使用 Xwindow，那么就使用 make menuconfig 好了。界面虽然比上面一个差点，总比 make config 的要好多了。

在这里我们输入命令：# make menuconfig

下图为 make menuconfig 的界面：

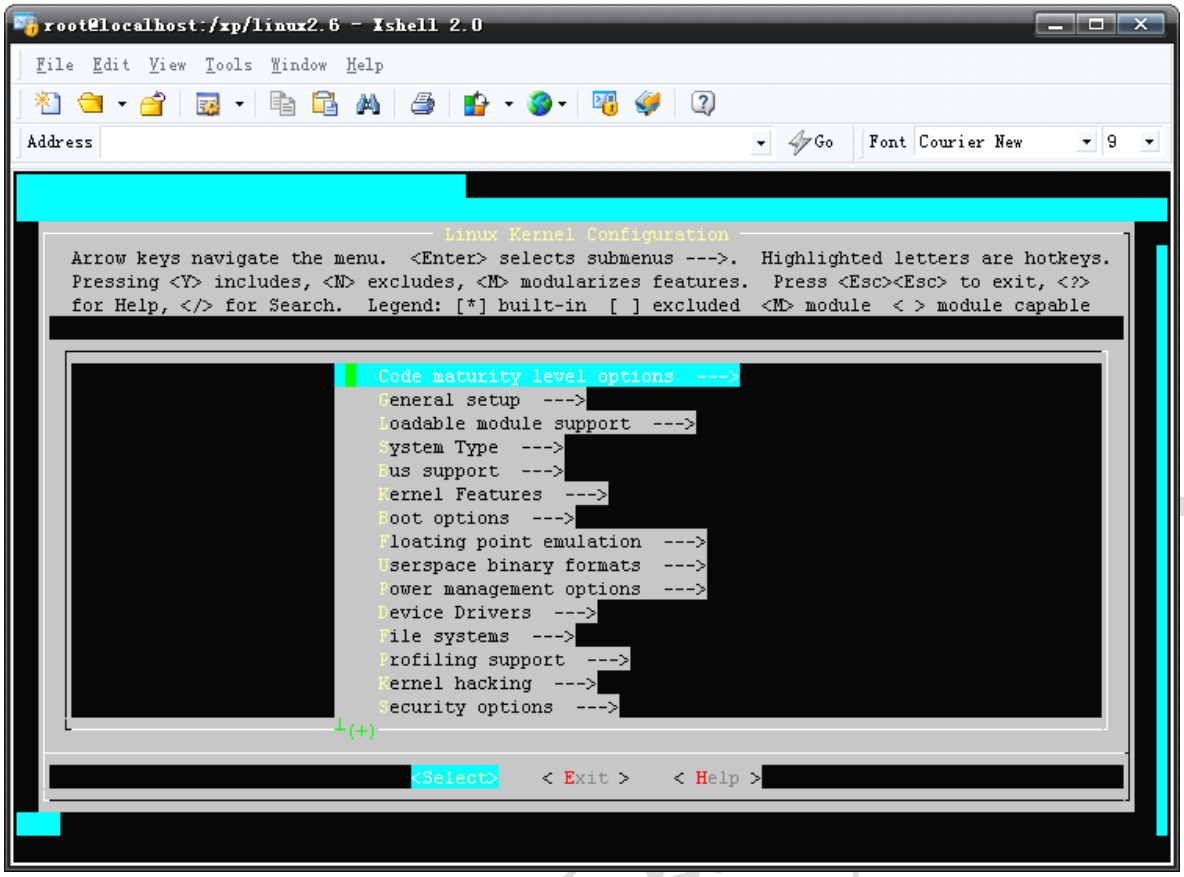
选择相应的配置时，有三种选择，它们分别代表的含义如下：

Y—-将该功能编译进内核

N—-不将该功能编译进内核

M—-将该功能编译成可以在需要时动态插入到内核中的模块

需要使用空格键进行选取。



进入配置栏的的“Load an Alternate Configuration File”

输入配置文件名 kernel_2440.cfg 并回车，在主菜单里选择 Exit 退出并保存设置。

4、输入命令 make dep

make dep 实际上读取配置过程生成的配置文件，来创建对应于配置的依赖关系树，从而决定哪些需要编译而那些不需要。

5、接着输入命令 make zImage

实现完全编译内核

6、编译结束后，可在 arch/arm/boot 目录下得到 linux 内核压缩映像 zImage。

生成内核映像后可以尝试烧写测试一下是否能运行，值得注意的是测试内核的前提是烧好了根文件系统，单独烧写内核映像是不能工作的。你可以用光盘中提供的现有的根文件系统进行测试，也可以自己做根文件系统。

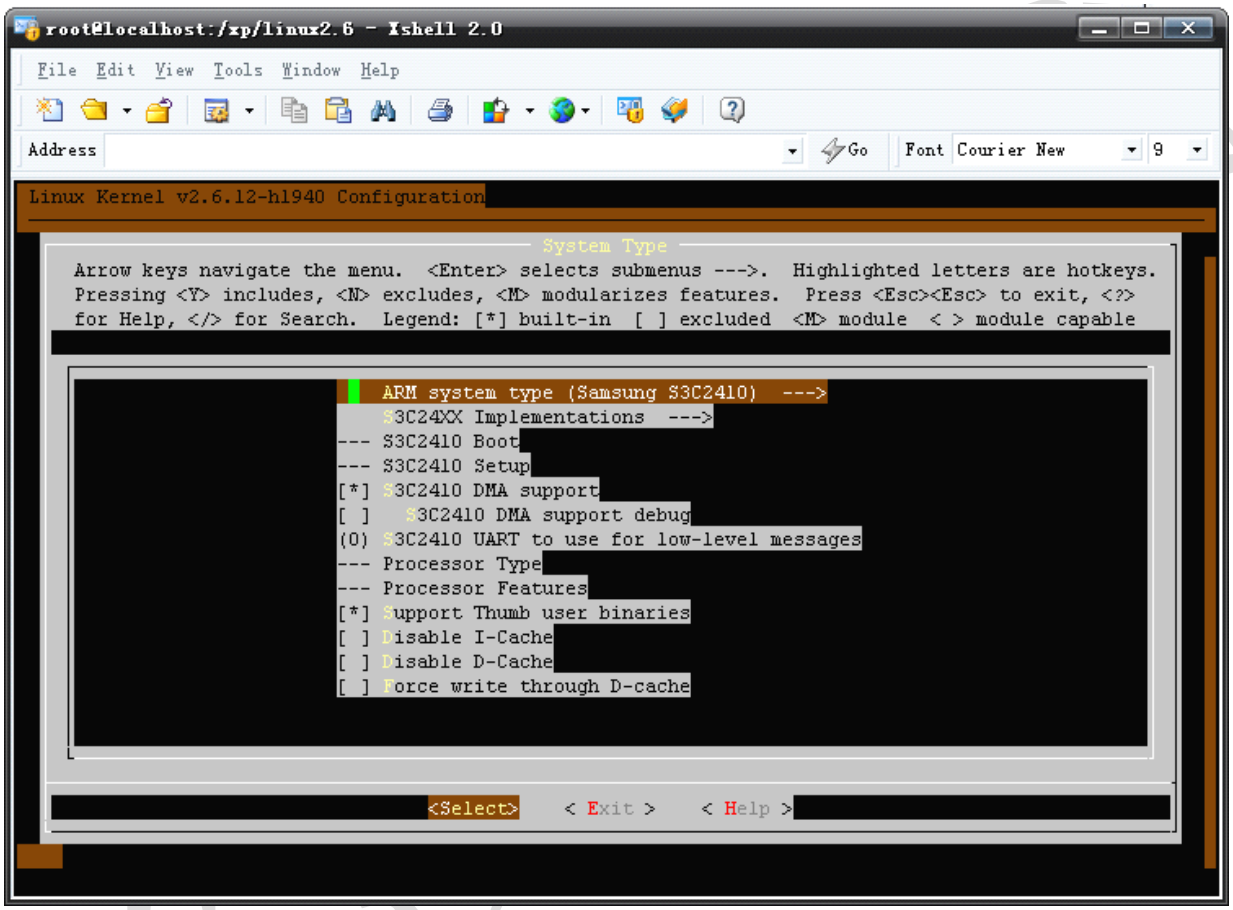
2.5 内核配置说明

上一节我们介绍了如何编译内核，其中配置内核部分我们用的是缺省的配置文件，除此之外我们也可以自己进行配置，下面就一些常见的配置进行说明。进入内核所在目录，输入 make menuconfig 进入配置

菜单。

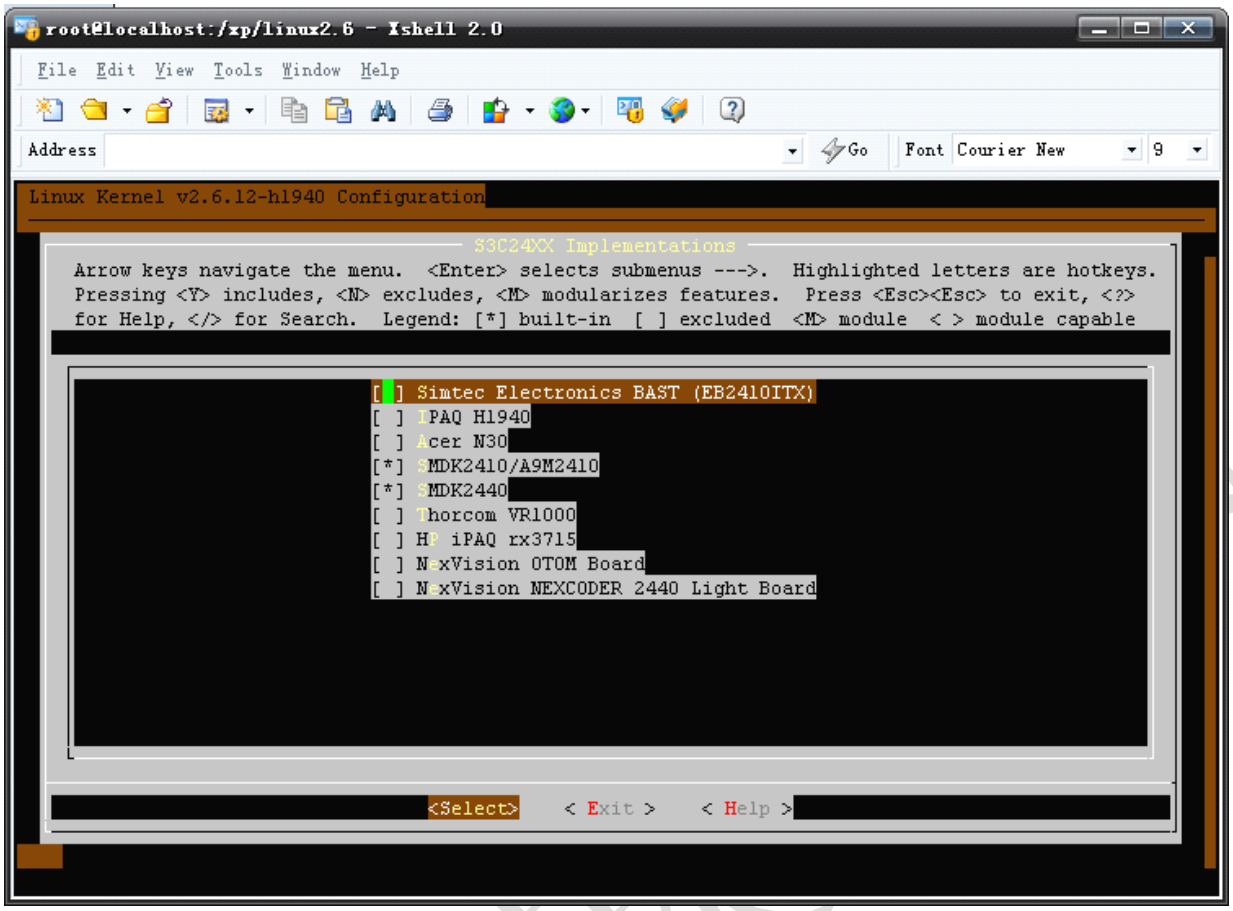
2.5.1 配置 CPU

在主菜单里面，选择 System Type，按回车进入



因为 S3C2410 和 S3C2440 的很多寄存器地址和设置是完全相同的，Linux-2.6 不再对这两种 CPU 进行区分，所以看到的很多配置都标注的是 S3C2410。

如果您要选择板级选项，可以进入 S3C24XX Implementations 子菜单，里面有很多常见的使用基于 S3C2410 和 S3C2440 的目标板平台选项。


root@localhost:/xp/linux2.6 - Xshell 2.0
File Edit View Tools Window Help
Address
Go
Font Courier New
9
Linux Kernel v2.6.12-hl940 Configuration
S3C24XX Implementations
Arrow keys navigate the menu. <Enter> selects submenus --->. Highlighted letters are hotkeys.
Pressing <Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?>
for Help, </> for Search. Legend: [*] built-in [ ] excluded <M> module < > module capable
[ ] Simtec Electronics BAST (EB2410ITX)
[ ] IPAQ H1940
[ ] Acer N30
[*] SMDK2410/A9M2410
[*] SMDK2440
[ ] Thorcom VR1000
[ ] HP iPAQ rx3715
[ ] NexVision OTOM Board
[ ] NexVision NEXCODER 2440 Light Board
<Select> < Exit > < Help >

2.5.2 配置 USB 鼠标和键盘

在 Device Drivers 找到 USB support，按回车进入

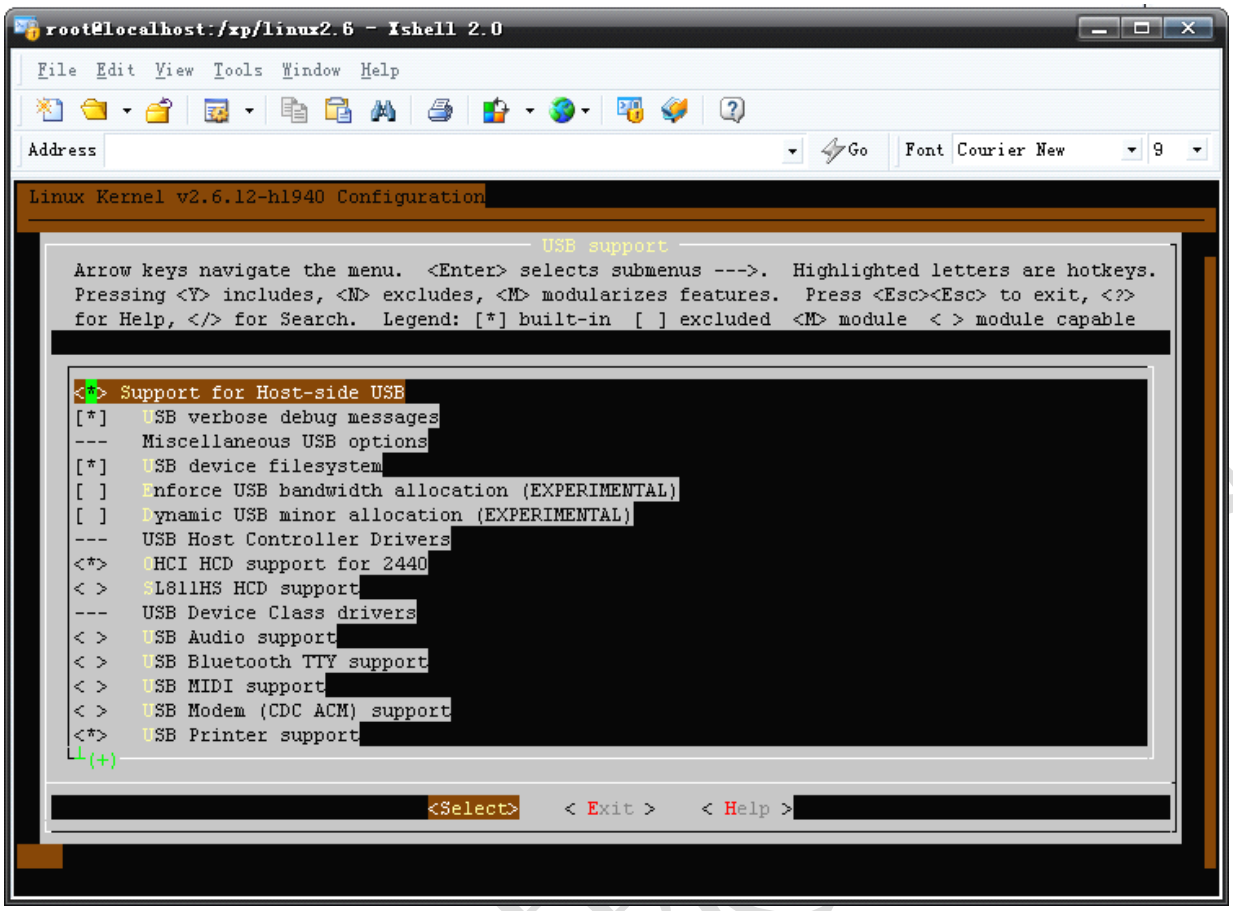
选中以下选项

<*> Support for Host-side USB

<*> OHCI HCD support

<*> USB Human Interface Devices (full HID) support

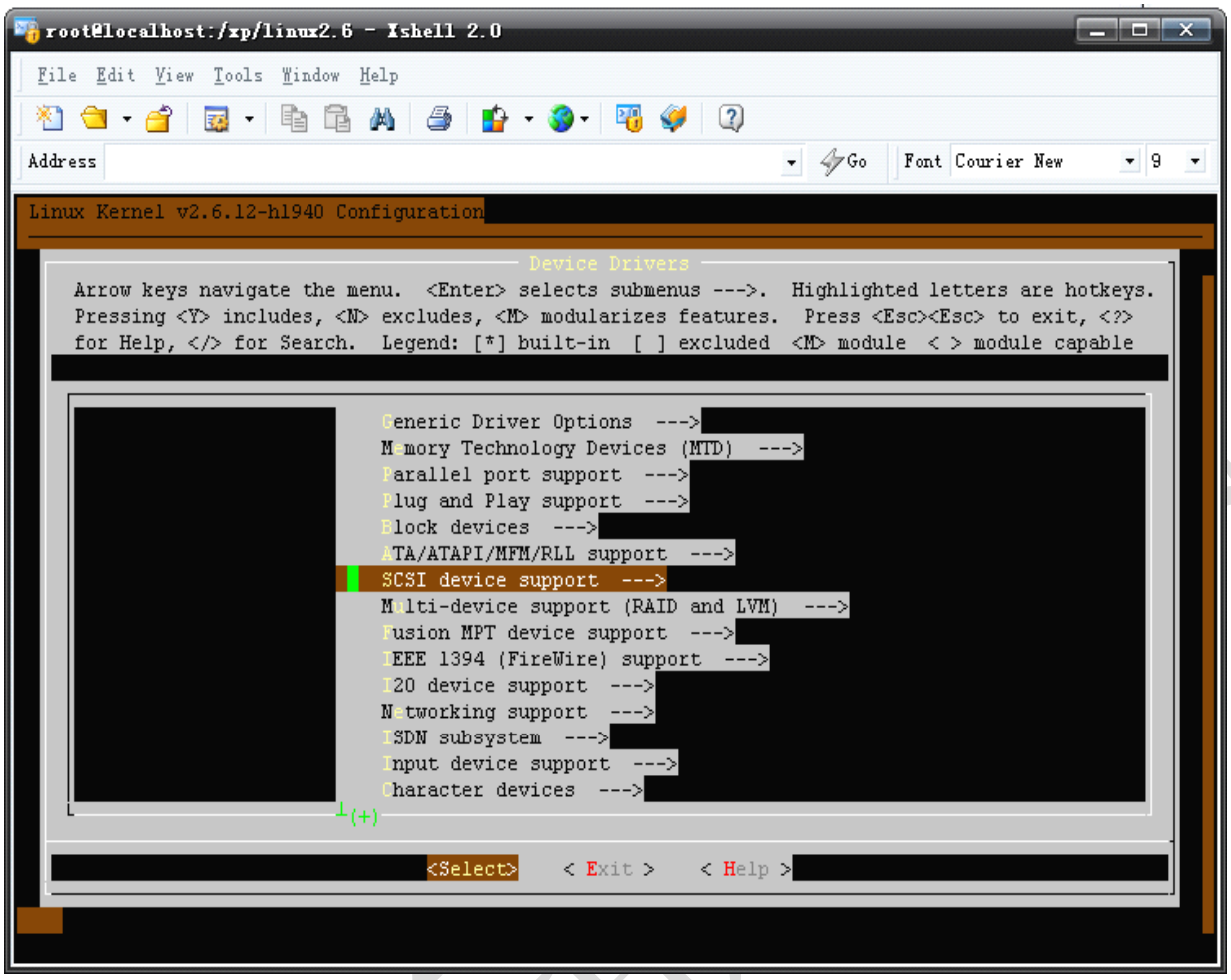
[*] HID input layer support


root@localhost:/xp/linux2.6 - Xshell 2.0
File Edit View Tools Window Help
Address
Go
Font Courier New
9
Linux Kernel v2.6.12-hl940 Configuration
USB support
Arrow keys navigate the menu. <Enter> selects submenus --->. Highlighted letters are hotkeys.
Pressing <Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?>
for Help, </> for Search. Legend: [*] built-in [ ] excluded <M> module < > module capable
<*> Support for Host-side USB
[*] USB verbose debug messages
--- Miscellaneous USB options
[*] USB device filesystem
[ ] Enforce USB bandwidth allocation (EXPERIMENTAL)
[ ] Dynamic USB minor allocation (EXPERIMENTAL)
--- USB Host Controller Drivers
<*> OHCI HCD support for 2440
< > SL811HS HCD support
--- USB Device Class drivers
< > USB Audio support
< > USB Bluetooth TTY support
< > USB MIDI support
< > USB Modem (CDC ACM) support
<*> USB Printer support
<Select> < Exit > < Help >

2.5.3 配置 U 盘支持

因为要优盘用到了SCSI 命令，所以我们先增加SCSI 支持。

在 Device Drivers 菜单里面，选择 SCSI device support，按回车进入

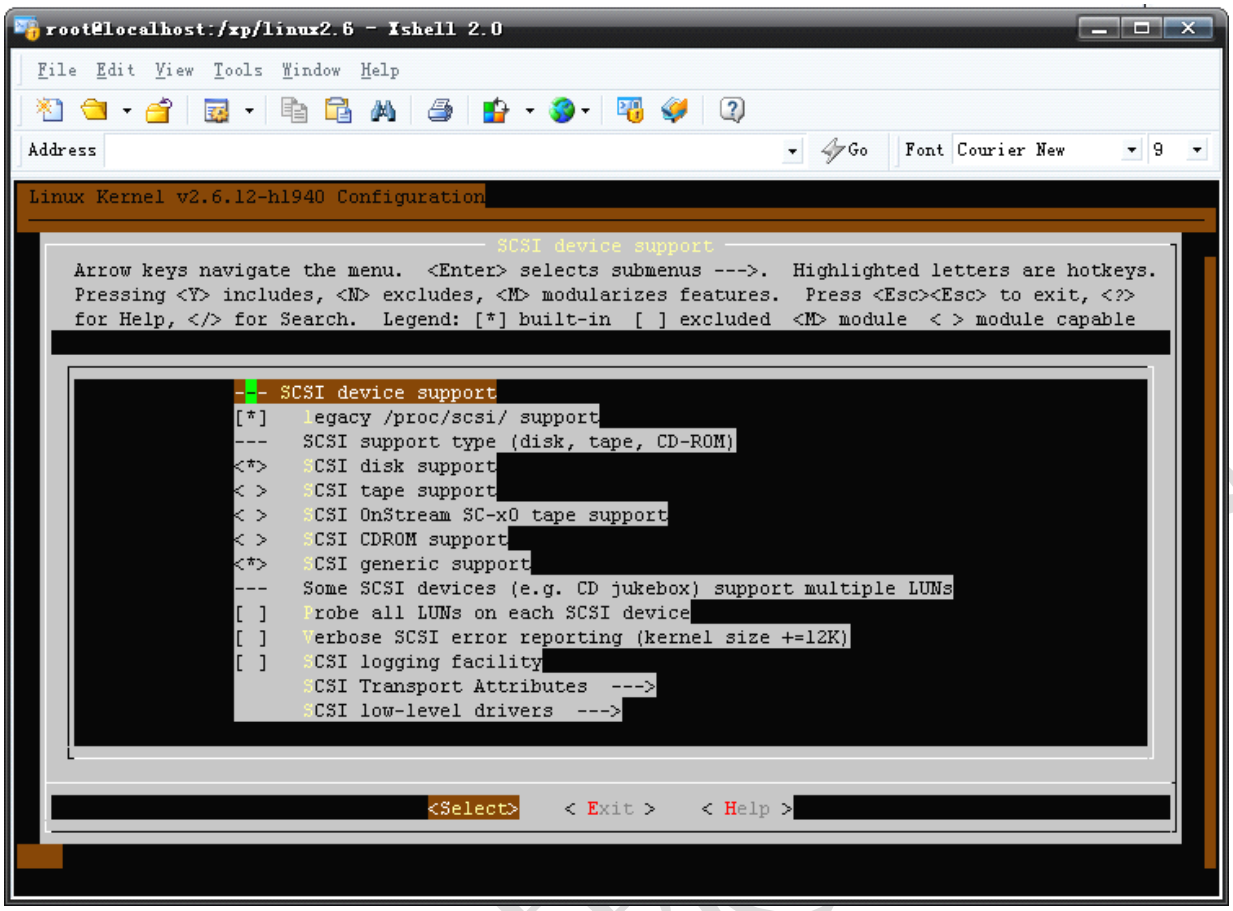

root@localhost:/xp/linux2.6 - Xshell 2.0
File Edit View Tools Window Help
Address
Go
Font Courier New
9
Linux Kernel v2.6.12-hl940 Configuration
Device Drivers
Arrow keys navigate the menu. <Enter> selects submenus --->. Highlighted letters are hotkeys.
Pressing <Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?>
for Help, </> for Search. Legend: [*] built-in [ ] excluded <M> module < > module capable
Generic Driver Options --->
Memory Technology Devices (MTD) --->
Parallel port support --->
Plug and Play support --->
Block devices --->
ATA/ATAPI/MFM/RLL support --->
SCSI device support --->
Multi-device support (RAID and LVM) --->
Fusion MPT device support --->
IEEE 1394 (FireWire) support --->
I2O device support --->
Networking support --->
ISDN subsystem --->
Input device support --->
Character devices --->
<Select> < Exit > < Help >

选中以下选项：

[*] legacy /proc/scsi support

<*> SCSI disk support

<*> SCSI generic support

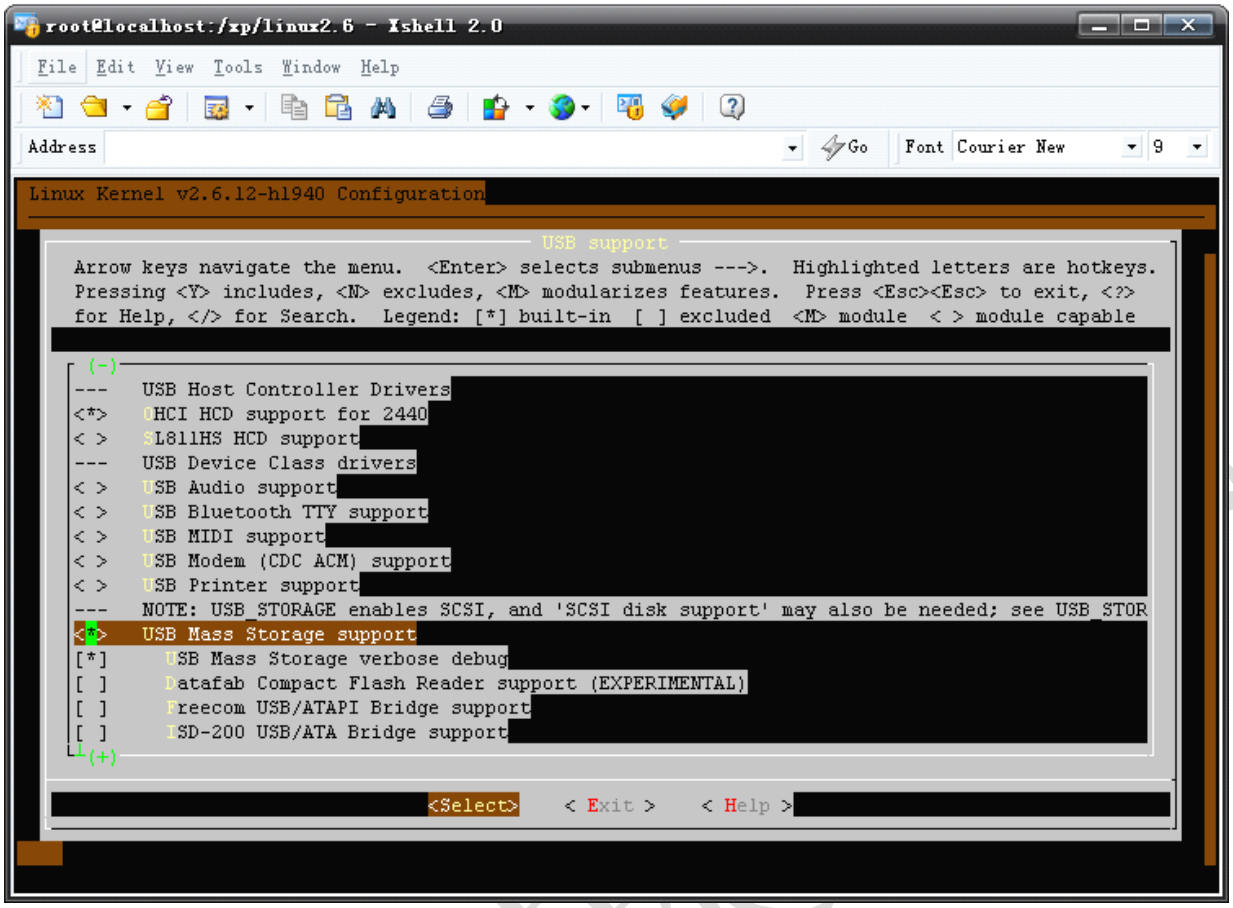

root@localhost:/xp/linux2.6 - Xshell 2.0
File Edit View Tools Window Help
Address
Go
Font Courier New
9
Linux Kernel v2.6.12-hl940 Configuration
SCSI device support
Arrow keys navigate the menu. <Enter> selects submenus --->. Highlighted letters are hotkeys.
Pressing <Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?>
for Help, </> for Search. Legend: [*] built-in [ ] excluded <M> module < > module capable
--- SCSI device support
[*] legacy /proc/scsi/ support
--- SCSI support type (disk, tape, CD-ROM)
<*> SCSI disk support
< > SCSI tape support
< > SCSI OnStream SC-x0 tape support
< > SCSI CDROM support
<*> SCSI generic support
--- Some SCSI devices (e.g. CD jukebox) support multiple LUNs
[ ] Probe all LUNs on each SCSI device
[ ] Verbose SCSI error reporting (kernel size +=12K)
[ ] SCSI logging facility
SCSI Transport Attributes --->
SCSI low-level drivers --->
<Select> < Exit > < Help >

选中Exit按回车返回到Device Driver菜单

选择USB support，按回车进入

选中以下选项

<*> USB Mass Storage support


root@localhost:/xp/linux2.6 - Xshell 2.0
File Edit View Tools Window Help
Address
Go
Font Courier New
9
Linux Kernel v2.6.12-hl940 Configuration
USB support
Arrow keys navigate the menu. <Enter> selects submenus --->. Highlighted letters are hotkeys.
Pressing <Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?>
for Help, </> for Search. Legend: [*] built-in [ ] excluded <M> module < > module capable
(-)
--- USB Host Controller Drivers
<*> OHCI HCD support for 2440
< > SL811HS HCD support
--- USB Device Class drivers
< > USB Audio support
< > USB Bluetooth TTY support
< > USB MIDI support
< > USB Modem (CDC ACM) support
< > USB Printer support
--- NOTE: USB_STORAGE enables SCSI, and 'SCSI disk support' may also be needed; see USB_STOR
<M> USB Mass Storage support
[*] USB Mass Storage verbose debug
[ ] Datafab Compact Flash Reader support (EXPERIMENTAL)
[ ] Freecom USB/ATAPI Bridge support
[ ] ISD-200 USB/ATA Bridge support
(+)
<Select> < Exit > < Help >

2.5.4 配置 SD/MMC 卡驱动

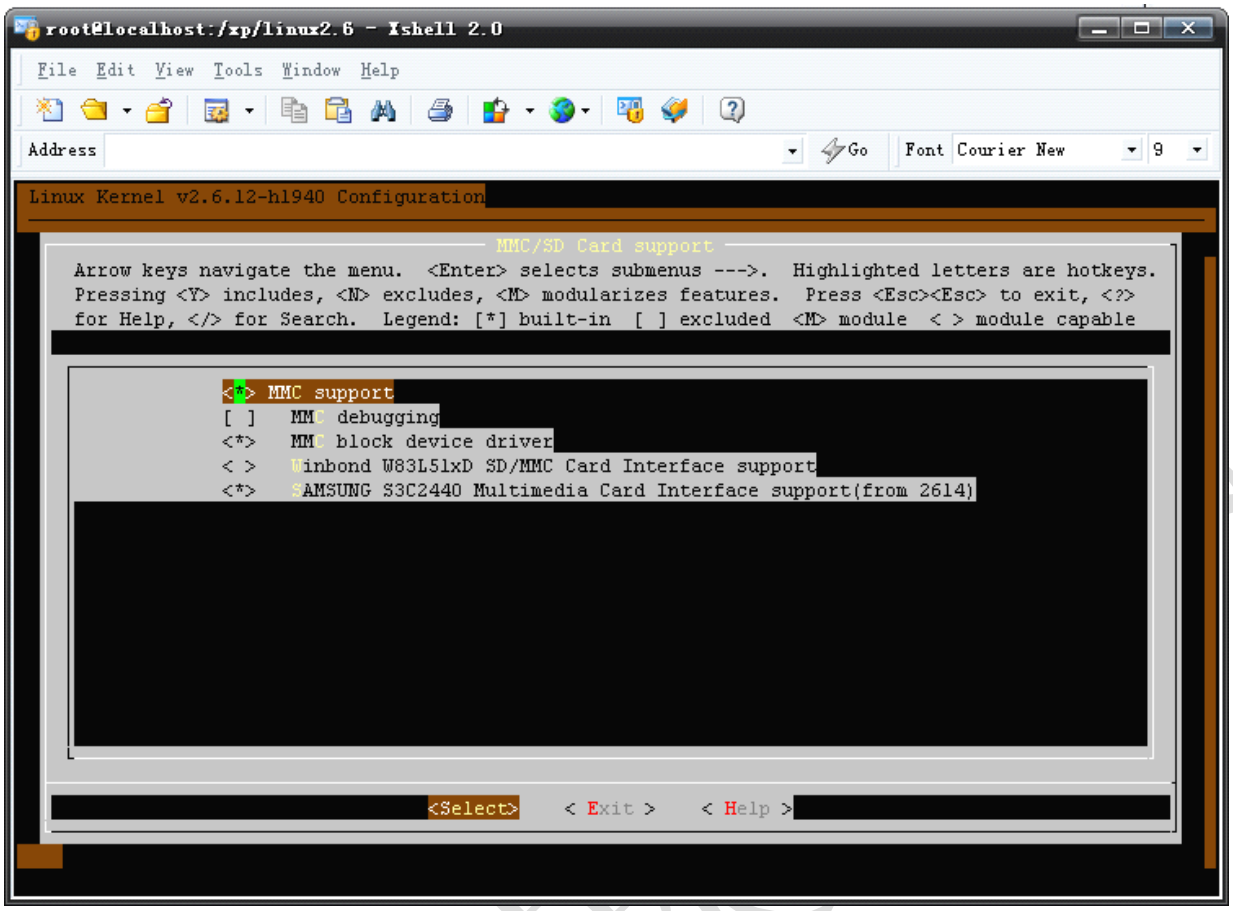
在 Device Drivers 菜单中，选择进入 MMC/SD Card support

选中以下选项

<*> MMC support

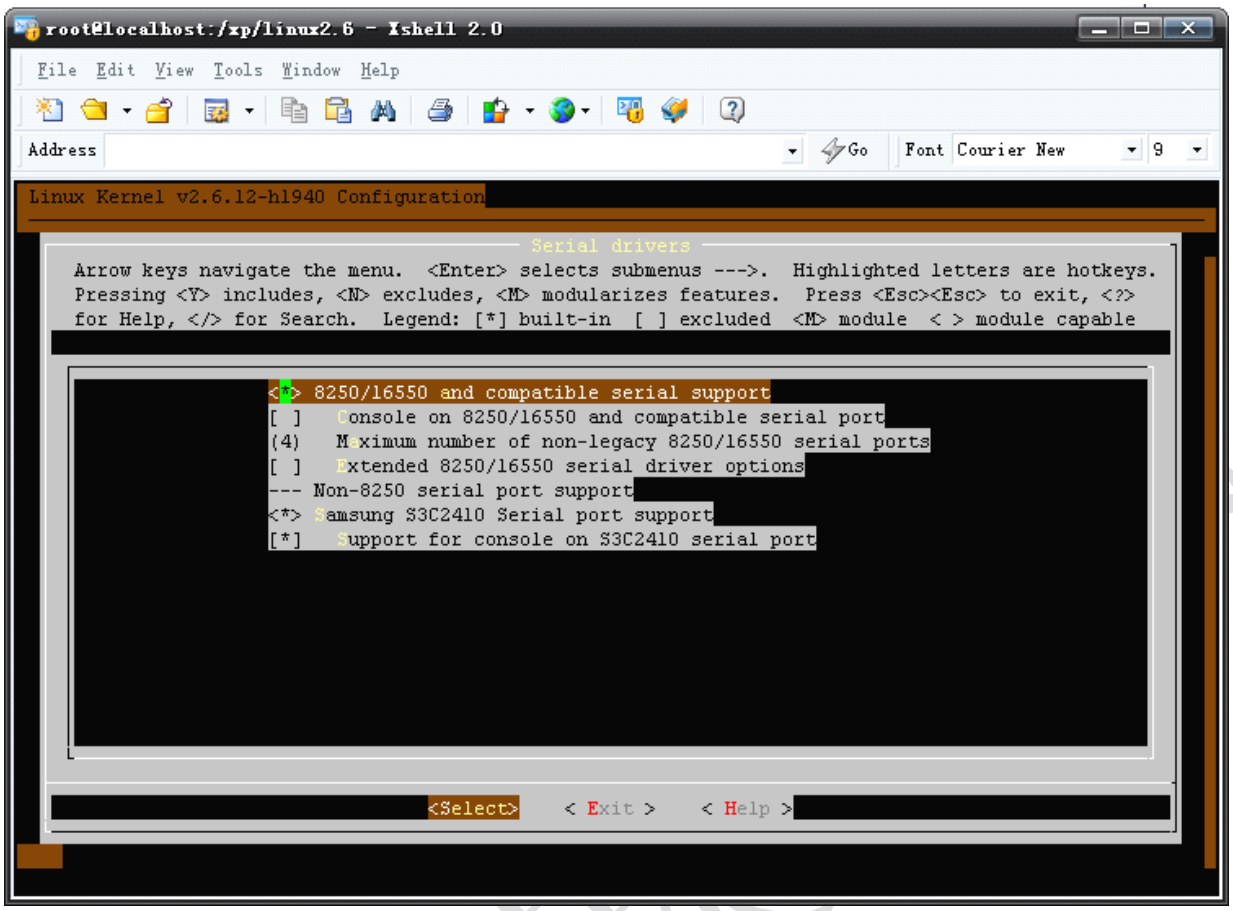
<*> MMC block device driver

<*> SAMSUNG S3C2440 Multimedia Card Interface support(from 2614)


root@localhost:/xp/linux2.6 - Xshell 2.0
File Edit View Tools Window Help
Address
Go
Font Courier New
9
Linux Kernel v2.6.12-hl940 Configuration
MMC/SD Card support
Arrow keys navigate the menu. <Enter> selects submenus --->. Highlighted letters are hotkeys.
Pressing <Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?>
for Help, </> for Search. Legend: [*] built-in [ ] excluded <M> module < > module capable
<*> MMC support
[ ] MMC debugging
<*> MMC block device driver
< > Winbond W83L51xD SD/MMC Card Interface support
<*> SAMSUNG S3C2440 Multimedia Card Interface support(from 2614)
<Select> < Exit > < Help >

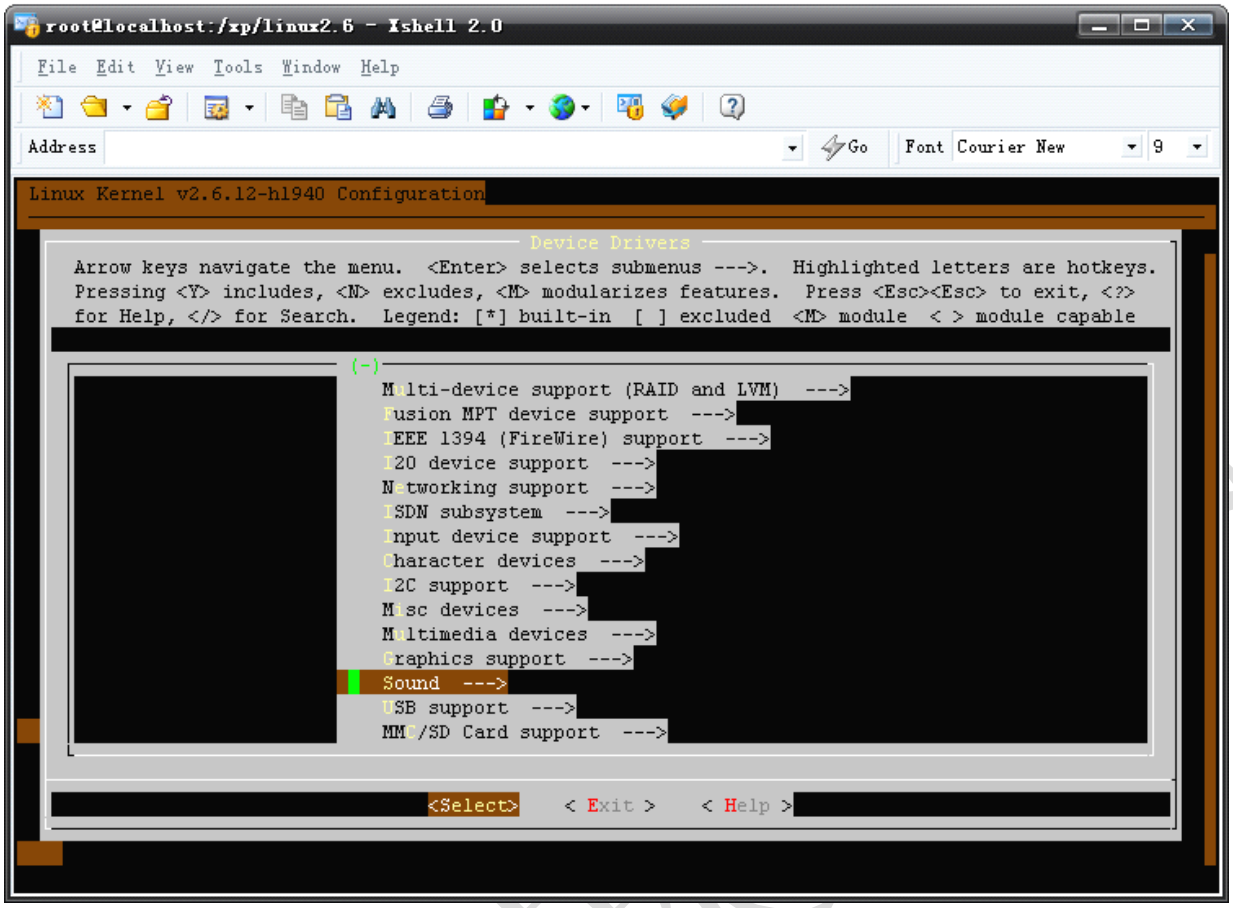
2.5.5 配置串口驱动

在 Device Drivers 菜单中，选择进入 Character devices - - ->，再选择进入 Serial drivers - - ->，如下图进行选择

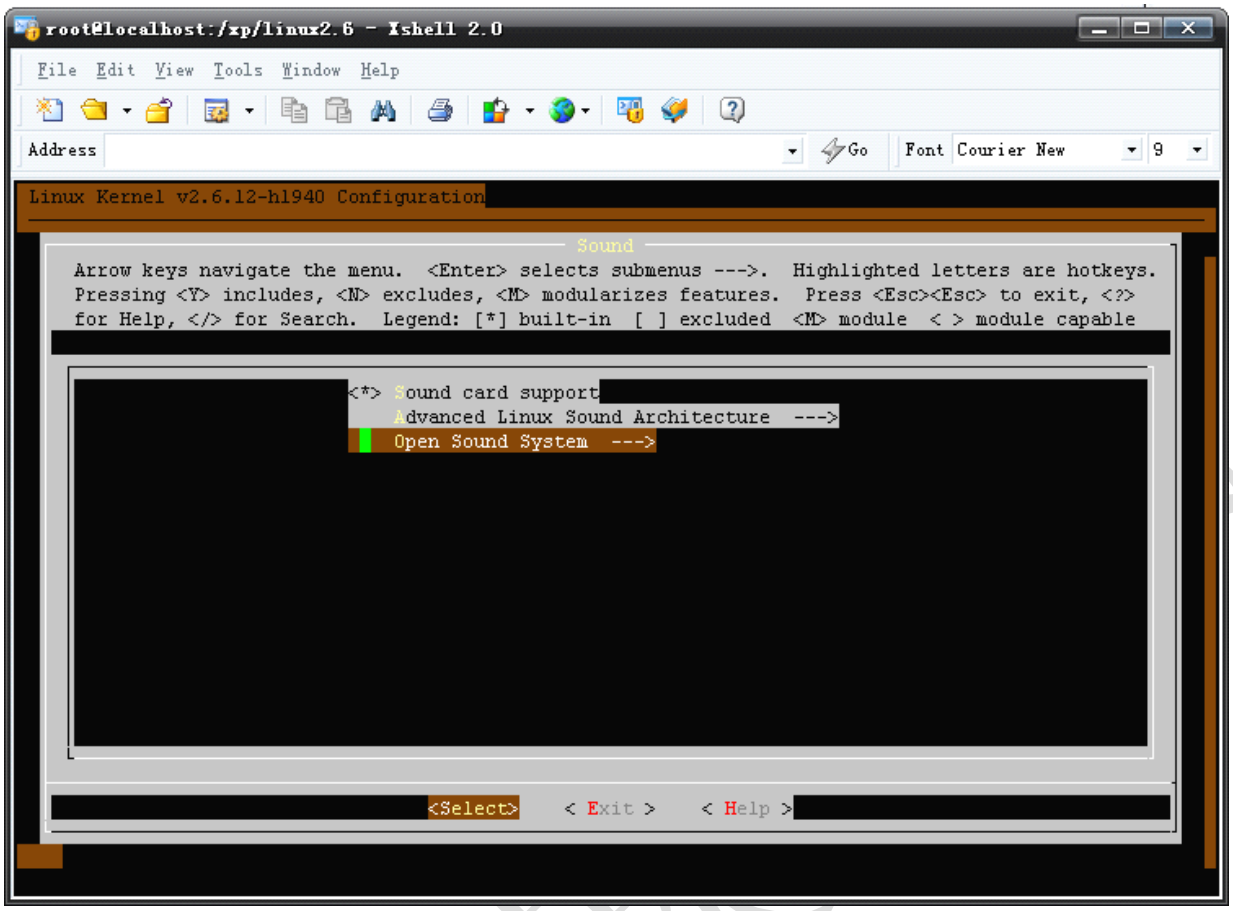

root@localhost:/xp/linux2.6 - Xshell 2.0
File Edit View Tools Window Help
Address
Go
Font Courier New
9
Linux Kernel v2.6.12-hl940 Configuration
Serial drivers
Arrow keys navigate the menu. <Enter> selects submenus --->. Highlighted letters are hotkeys.
Pressing <Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?>
for Help, </> for Search. Legend: [*] built-in [ ] excluded <M> module < > module capable
<*> 8250/16550 and compatible serial support
[ ] Console on 8250/16550 and compatible serial port
(4) Maximum number of non-legacy 8250/16550 serial ports
[ ] Extended 8250/16550 serial driver options
--- Non-8250 serial port support
<*> Samsung S3C2410 Serial port support
[*] Support for console on S3C2410 serial port
<Select> < Exit > < Help >

2.5.6 配置声卡驱动

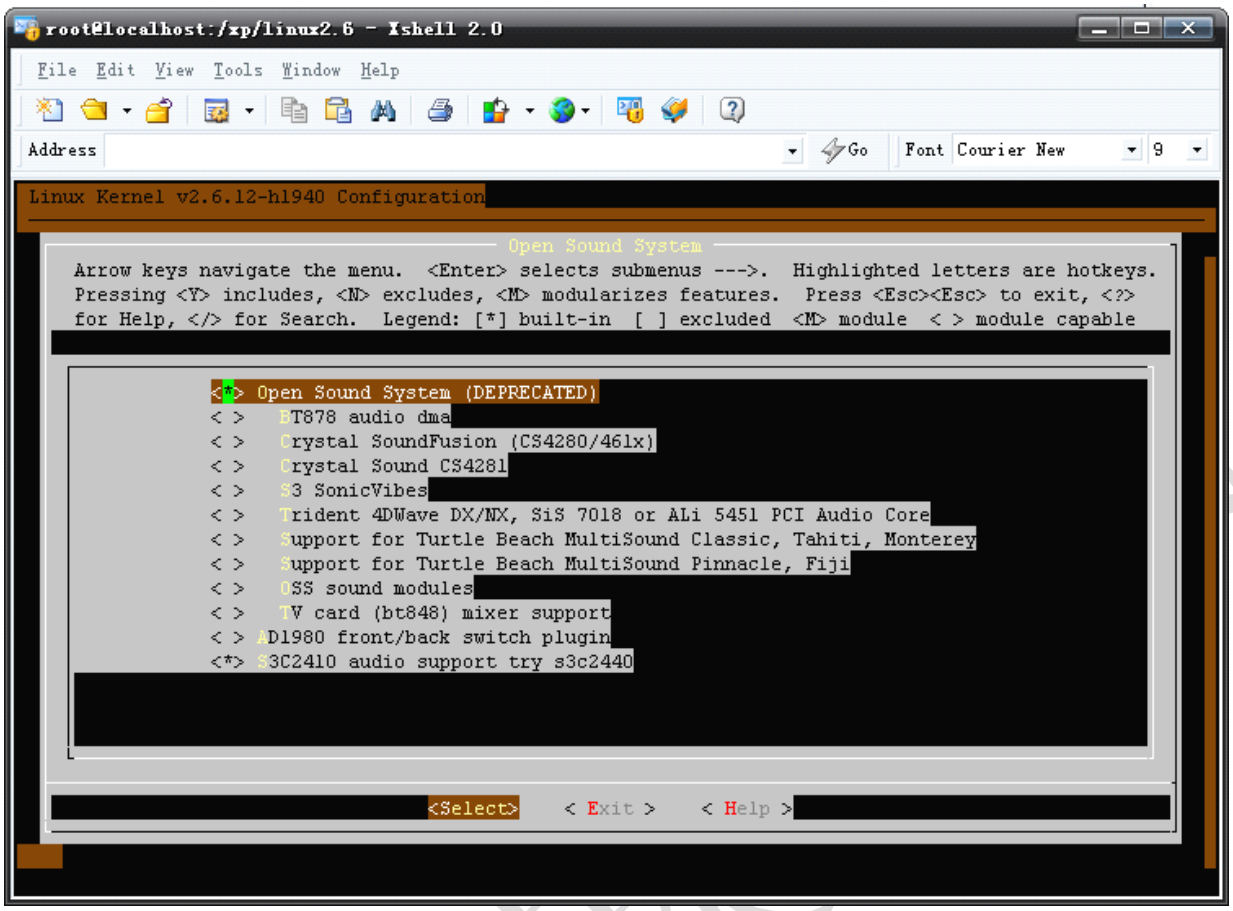
在 Device Drivers 菜单中，选择并进入 Sound - - ->


root@localhost:/xp/linux2.6 - Xshell 2.0
File Edit View Tools Window Help
Address
Go
Font Courier New
9
Linux Kernel v2.6.12-hl940 Configuration
Device Drivers
Arrow keys navigate the menu. <Enter> selects submenus --->. Highlighted letters are hotkeys.
Pressing <Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?>
for Help, </> for Search. Legend: [*] built-in [ ] excluded <M> module < > module capable
(-)
Multi-device support (RAID and LVM) --->
Fusion MPT device support --->
IEEE 1394 (FireWire) support --->
I2O device support --->
Networking support --->
ISDN subsystem --->
Input device support --->
Character devices --->
I2C support --->
Misc devices --->
Multimedia devices --->
Graphics support --->
Sound --->
USB support --->
MMC/SD Card support --->
<Select> < Exit > < Help >

选择进入Open Sound System - - ->

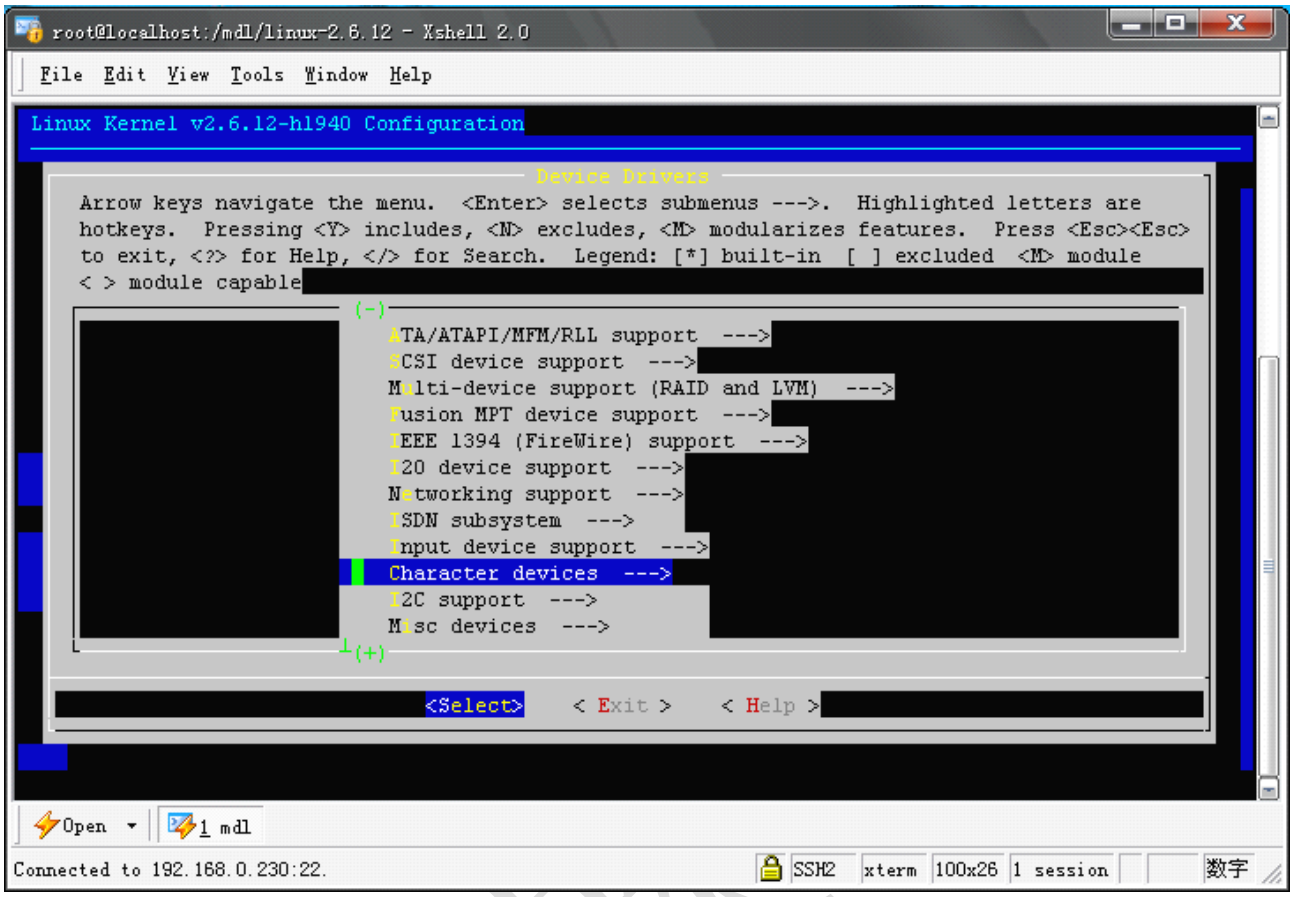

root@localhost:/xp/linux2.6 - Xshell 2.0
File Edit View Tools Window Help
Address
Go
Font Courier New
9
Linux Kernel v2.6.12-hl940 Configuration
Sound
Arrow keys navigate the menu. <Enter> selects submenus --->. Highlighted letters are hotkeys.
Pressing <Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?>
for Help, </> for Search. Legend: [*] built-in [ ] excluded <M> module < > module capable
<*> Sound card support
Advanced Linux Sound Architecture --->
Open Sound System --->
<Select> < Exit > < Help >

选择如图所示选项

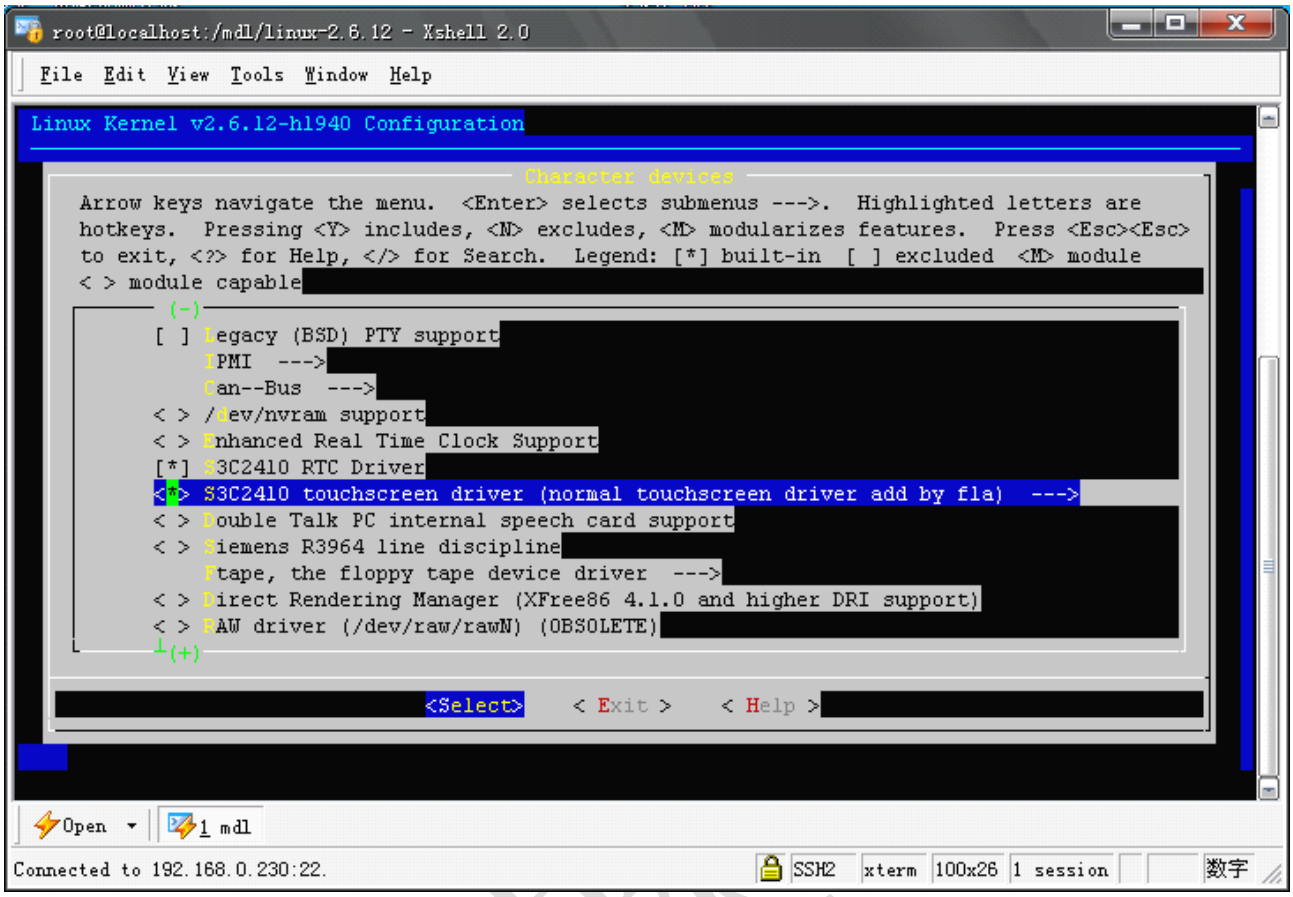


2.5.7 如何配置触摸屏

在配置菜单中选择并进入Device Drivers - - ->

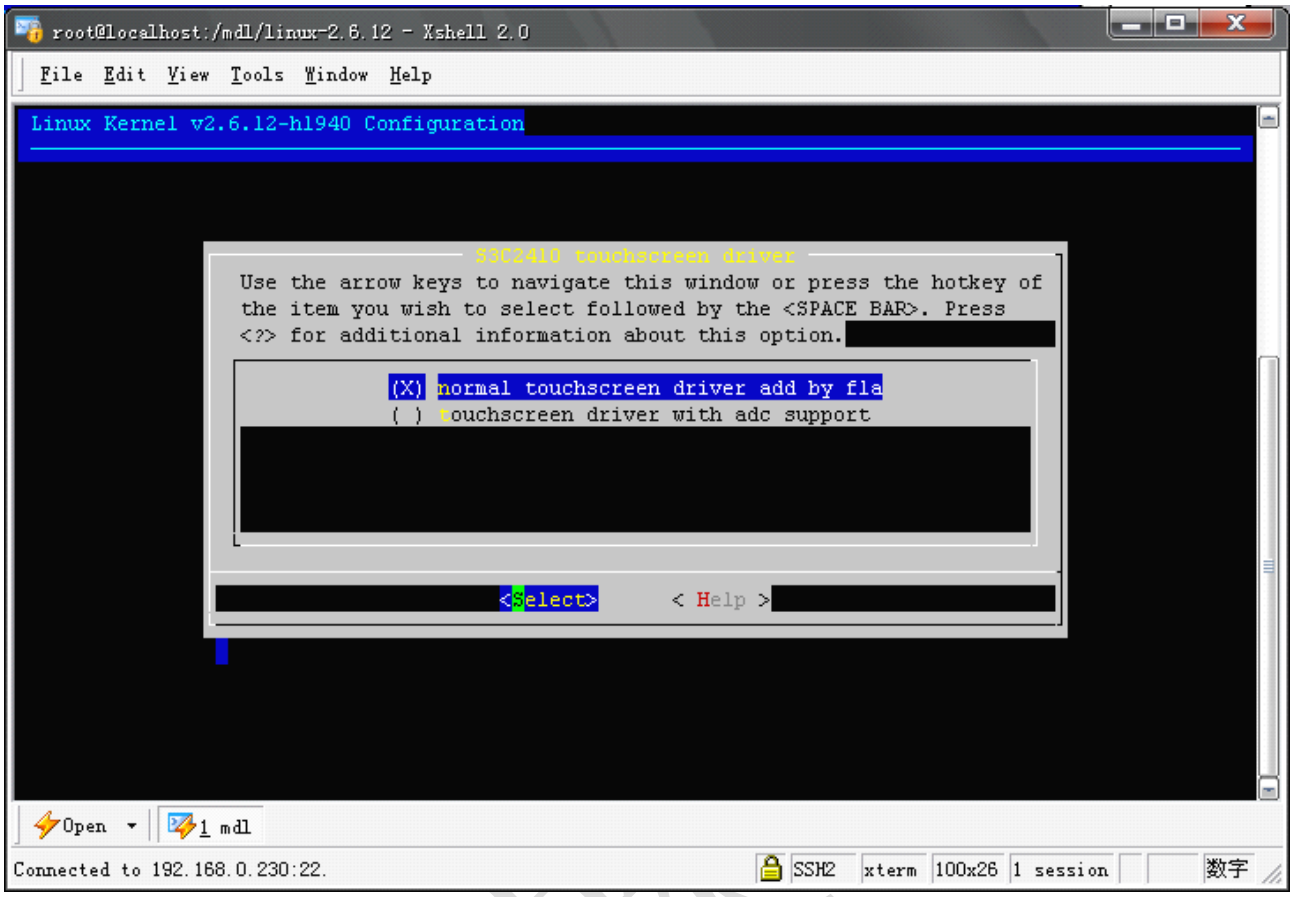

root@localhost:/mdl/linux-2.6.12 - Xshell 2.0
File Edit View Tools Window Help
Linux Kernel v2.6.12-hl940 Configuration
Device Drivers
Arrow keys navigate the menu. <Enter> selects submenus --->. Highlighted letters are
hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc>
to exit, <?> for Help, </> for Search. Legend: [*] built-in [ ] excluded <M> module
< > module capable
(-)
ATA/ATAPI/MFM/RLL support --->
SCSI device support --->
Multi-device support (RAID and LVM) --->
Fusion MPT device support --->
IEEE 1394 (FireWire) support --->
I2O device support --->
Networking support --->
ISDN subsystem --->
Input device support --->
Character devices --->
I2C support --->
Misc devices --->
(+)
<Select> < Exit > < Help >
Open
1 mdl
Connected to 192.168.0.230:22.
SSH2 xterm 100x26 1 session
数字

选择并进入 character devices-----→


root@localhost:/mdl/linux-2.6.12 - Xshell 2.0
File Edit View Tools Window Help
Linux Kernel v2.6.12-hl940 Configuration
Character devices
Arrow keys navigate the menu. <Enter> selects submenus --->. Highlighted letters are
hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc>
to exit, <?> for Help, </> for Search. Legend: [*] built-in [ ] excluded <M> module
< > module capable
[ ] Legacy (BSD) PTY support
IPMI --->
Can--Bus --->
< > /dev/nvram support
< > Enhanced Real Time Clock Support
[*] S3C2410 RTC Driver
<*> S3C2410 touchscreen driver (normal touchscreen driver add by fla) --->
< > Double Talk PC internal speech card support
< > Siemens R3964 line discipline
Ftape, the floppy tape device driver --->
< > Direct Rendering Manager (XFree86 4.1.0 and higher DRI support)
< > RAW driver (/dev/raw/rawN) (OBSOLETE)
<Select> < Exit > < Help >
Open
1 mdl
Connected to 192.168.0.230:22.
SSH2 xterm 100x26 1 session
数字

按空格选上 <*> S3C2410 touchscreen driver (normal touchscreen driver add by fla) ---> 并进入

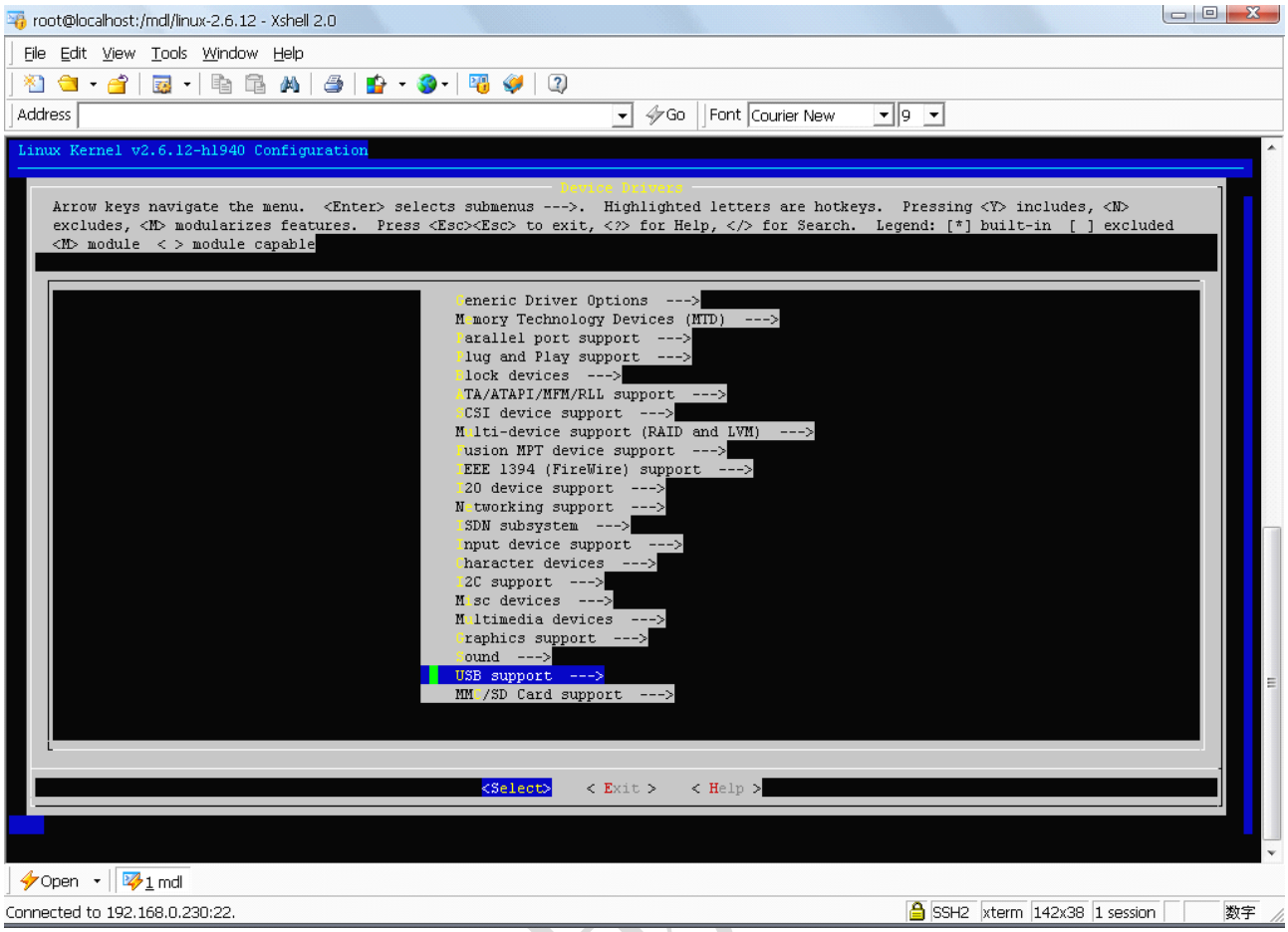
最终选项如下图所示



最后保存退出。

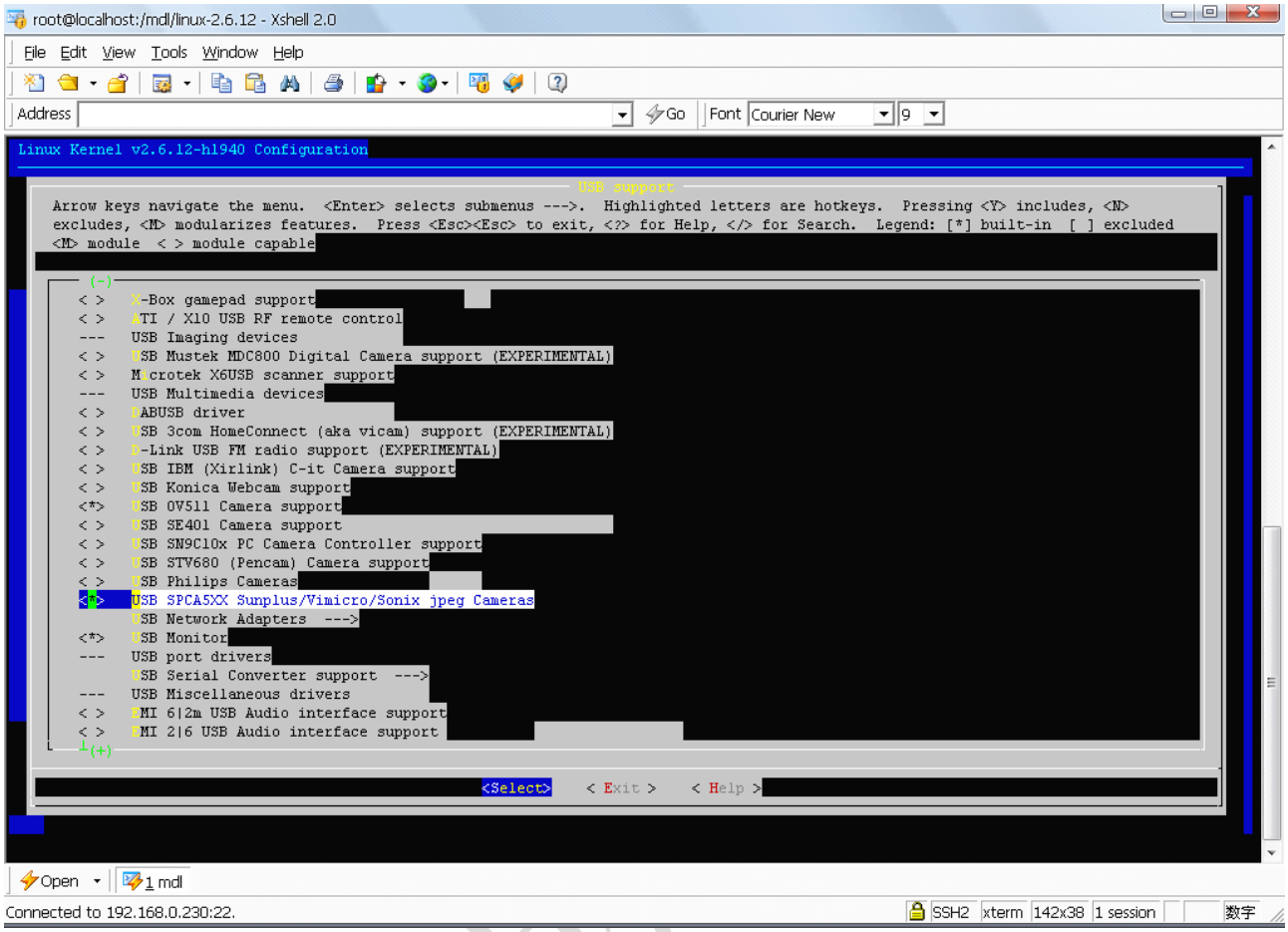
2.5.8 如何配置网眼和中芯微等 USB 摄像头

在配置菜单中选择并进入 Device Drivers - - ->


root@localhost:/mdl/linux-2.6.12 - Xshell 2.0
File Edit View Tools Window Help
Address
Go
Font Courier New 9
Linux Kernel v2.6.12-hl940 Configuration
Device Drivers
Arrow keys navigate the menu. <Enter> selects submenus --->. Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [*] built-in [ ] excluded <M> module < > module capable
Generic Driver Options --->
Memory Technology Devices (MTD) --->
Parallel port support --->
Plug and Play support --->
Block devices --->
ATA/ATAPI/MFM/RLL support --->
SCSI device support --->
Multi-device support (RAID and LVM) --->
Fusion MPT device support --->
IEEE 1394 (FireWire) support --->
I2O device support --->
Networking support --->
ISDN subsystem --->
Input device support --->
Character devices --->
I2C support --->
Misc devices --->
Multimedia devices --->
Graphics support --->
Sound --->
USB support --->
MMC/SD Card support --->
<Select> < Exit > < Help >
Open
1 mdl
Connected to 192.168.0.230:22.
SSH2 xterm 142x38 1 session
数字

然后选择并进入 USB support - - ->

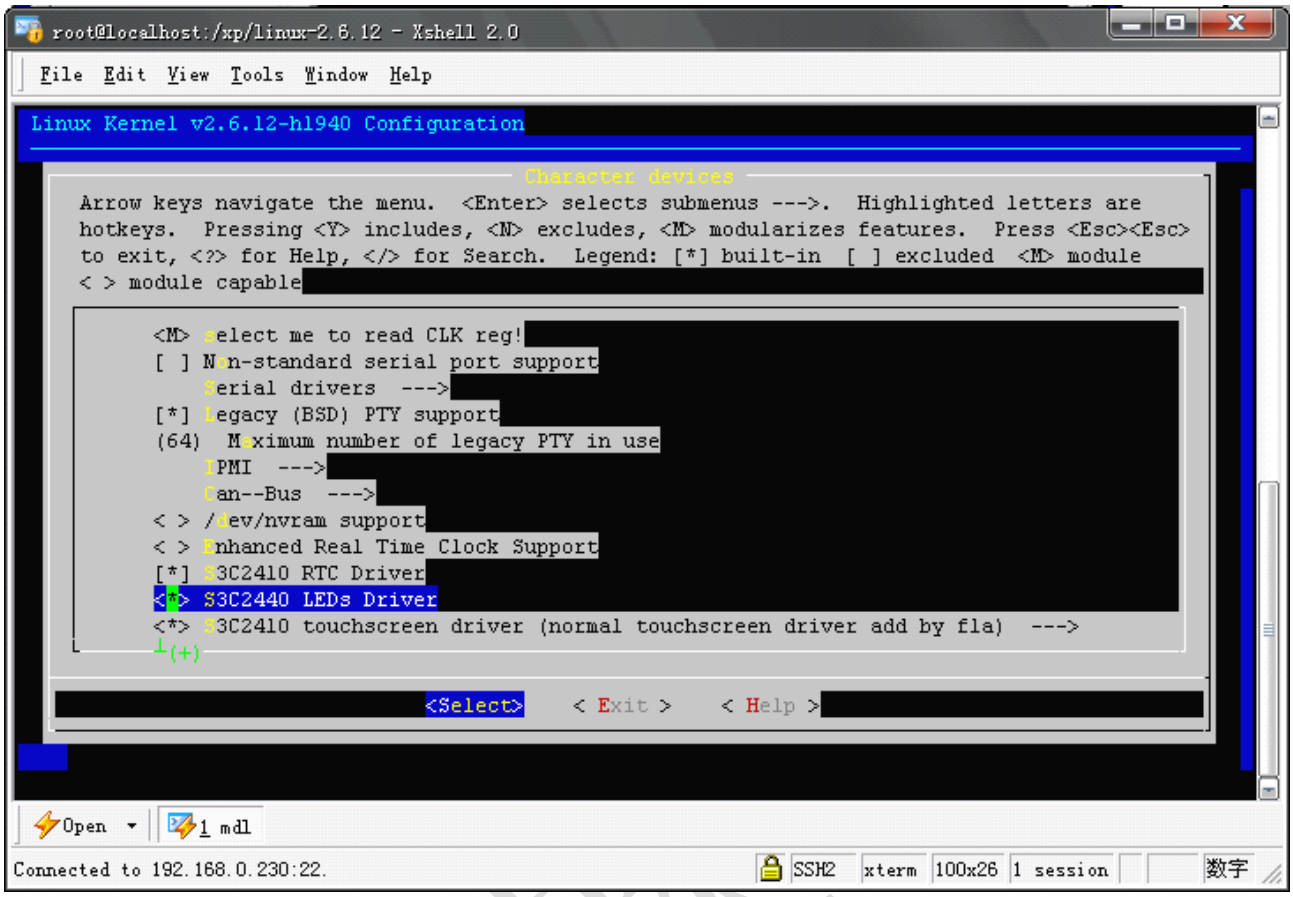
选上<*> USB SPCA5XX Sunplus/Vimicro/Sonix jpeg Cameras



2.5.9 如何配置 LED 驱动

在配置菜单中选择并进入 Device Drivers - - ->

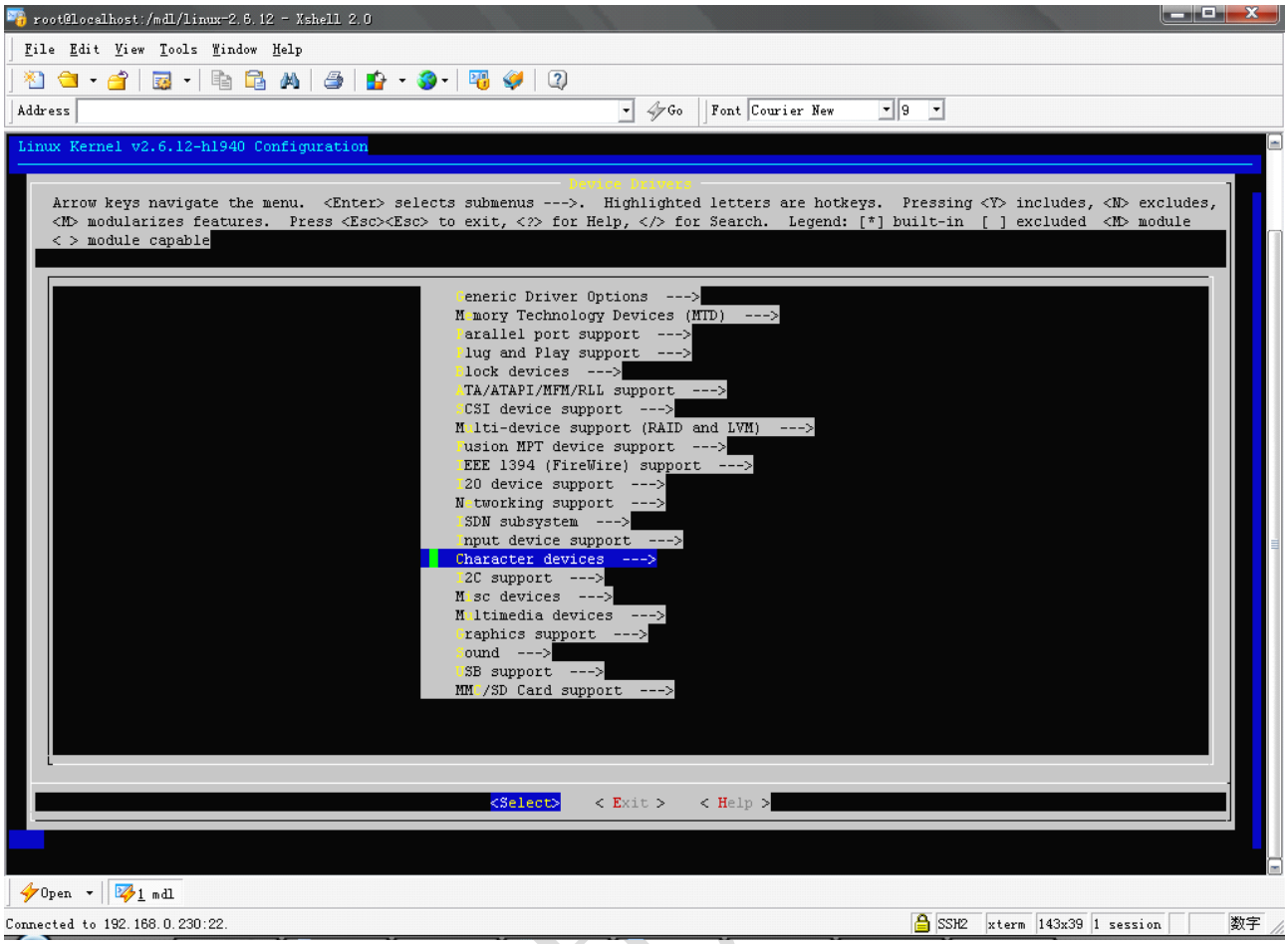
选择并进入 character devices-----→（以上步骤同配置触摸屏）


root@localhost:/xp/linux-2.6.12 - Xshell 2.0
File Edit View Tools Window Help
Linux Kernel v2.6.12-hl940 Configuration
Character devices
Arrow keys navigate the menu. <Enter> selects submenus --->. Highlighted letters are
hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc>
to exit, <?> for Help, </> for Search. Legend: [*] built-in [ ] excluded <M> module
< > module capable
<M> select me to read CLK reg!
[ ] Non-standard serial port support
Serial drivers --->
[*] Legacy (BSD) PTY support
(64) Maximum number of legacy PTY in use
IPMI --->
Can--Bus --->
< > /dev/nvram support
< > Enhanced Real Time Clock Support
[*] S3C2410 RTC Driver
<*> S3C2440 LEDs Driver
<*> S3C2410 touchscreen driver (normal touchscreen driver add by fla) --->
(+)
<Select> < Exit > < Help >
Open
1 mdl
Connected to 192.168.0.230:22.
SSH2 xterm 100x26 1 session
数字

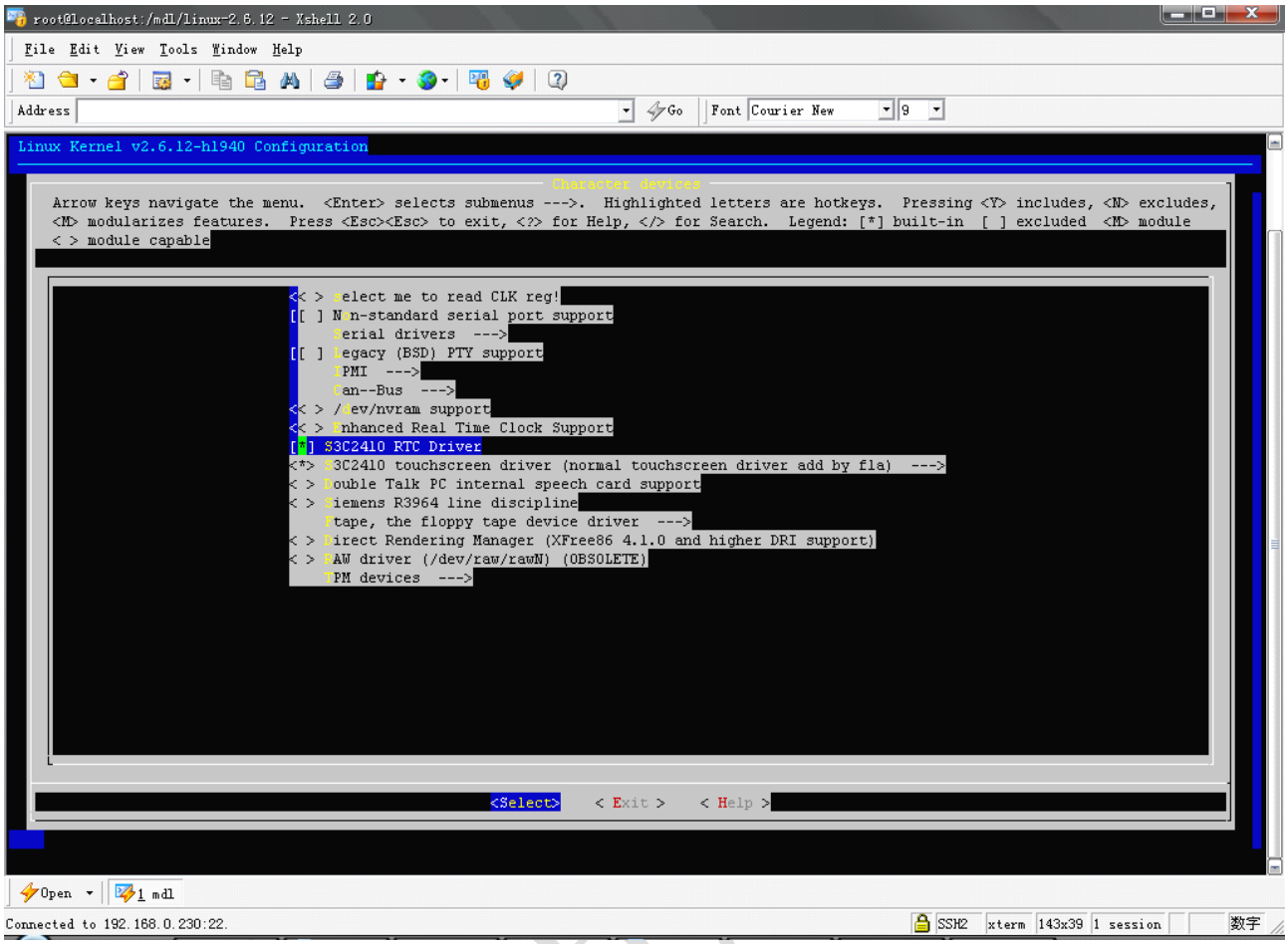
选择 <*> S3C2440 LEDs Driver 保存退出。

2.5.10 如何配置 RTC 实时时钟驱动

在配置菜单中选择并进入 Device Drivers - - ->



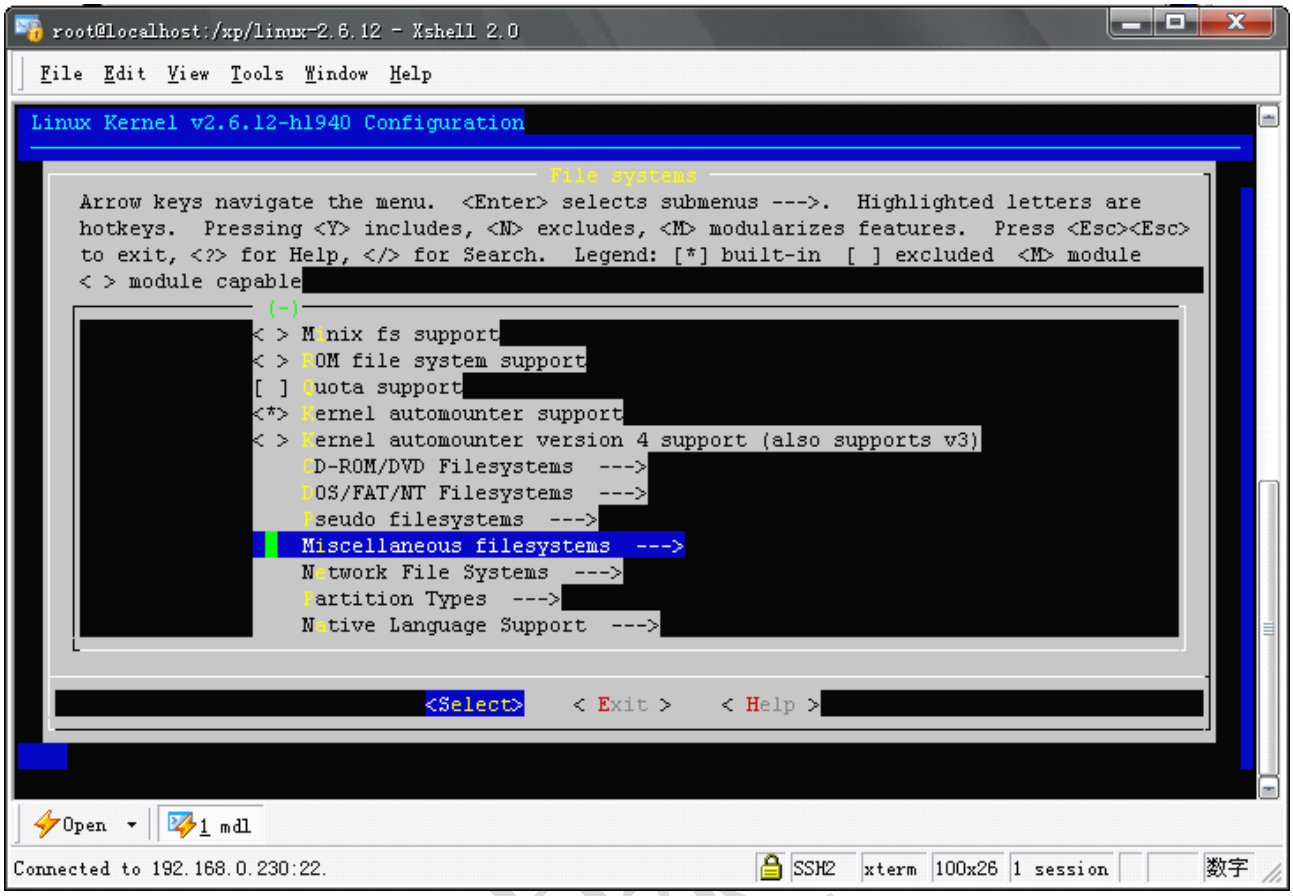
选择并进入 Character devices ---->



再次页面选择上[*] S3C2410 RTC Driver　然后保存退出即可。

2.5.11 如何配置 yaffs 文件系统的支持

选择并进入 File systems 菜单


root@localhost:/xp/linux-2.6.12 - Xshell 2.0
File Edit View Tools Window Help
Linux Kernel v2.6.12-hl940 Configuration
File systems
Arrow keys navigate the menu. <Enter> selects submenus --->. Highlighted letters are
hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc>
to exit, <?> for Help, </> for Search. Legend: [*] built-in [ ] excluded <M> module
< > module capable
(-)
< > Minix fs support
< > ROM file system support
[ ] Quota support
<*> Kernel automounter support
< > Kernel automounter version 4 support (also supports v3)
CD-ROM/DVD Filesystems --->
DOS/FAT/NT Filesystems --->
Pseudo filesystems --->
Miscellaneous filesystems --->
Network File Systems --->
Partition Types --->
Native Language Support --->
<Select> < Exit > < Help >
Open
1 mdl
Connected to 192.168.0.230:22.
SSH2 xterm 100x26 1 session
数字

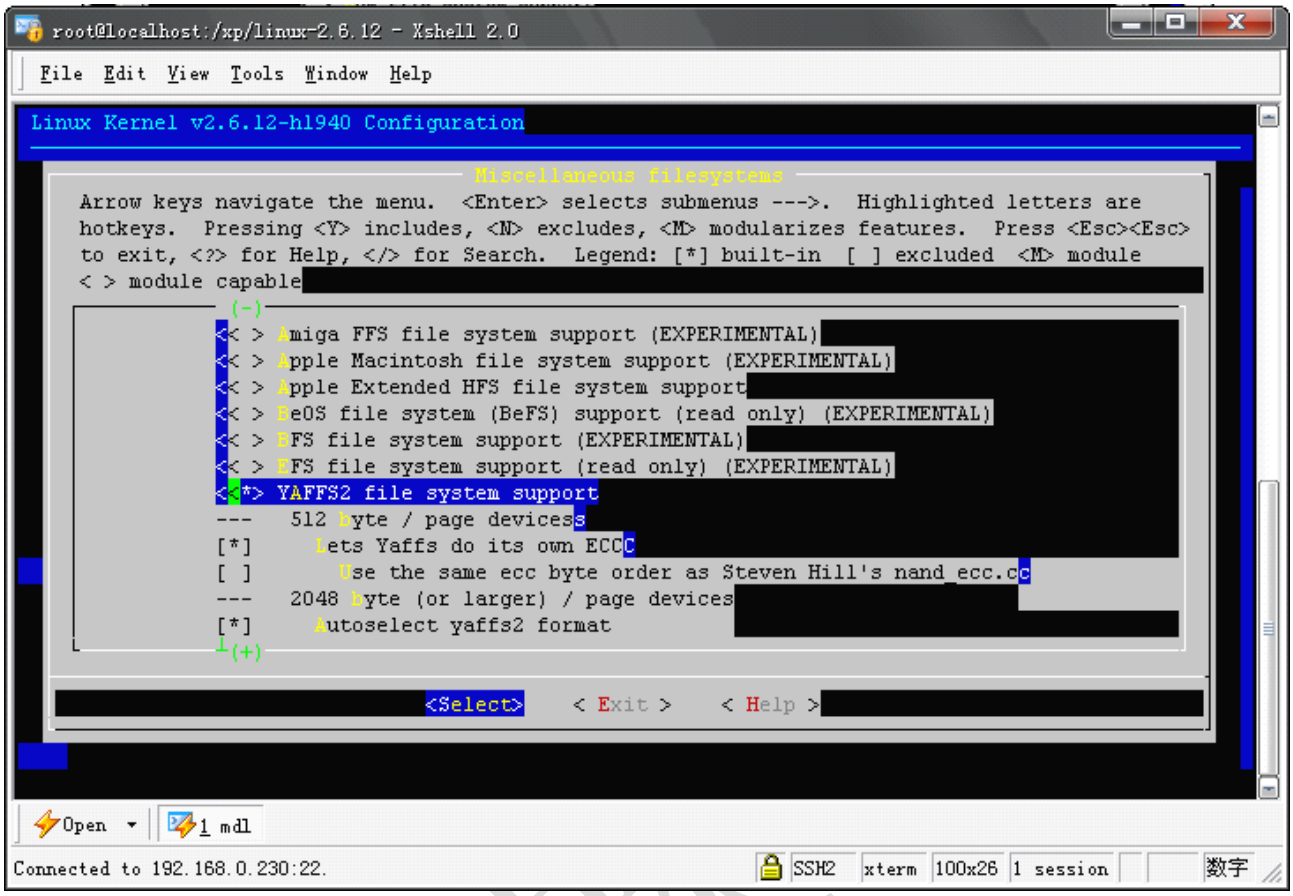
进入改选项，作如图所示选择：

<<*> YAFFS2 file system support

--- 512 byte / page devicess

[*] Lets Yaffs do its own ECCC

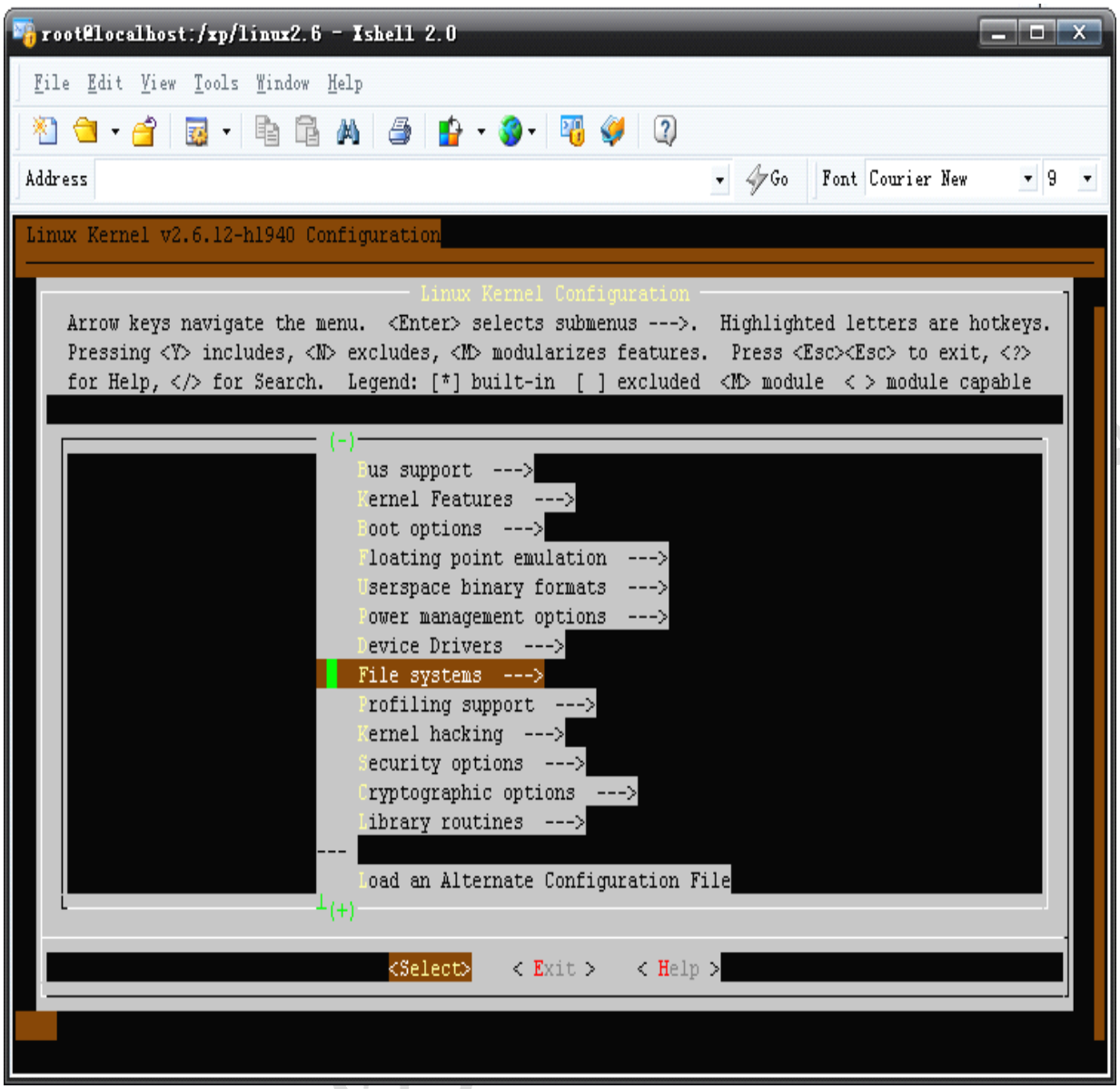
[ ] Use the same ecc byte order as Steven Hill's nand_ecc.cc


root@localhost:/xp/linux-2.6.12 - Xshell 2.0
File Edit View Tools Window Help
Linux Kernel v2.6.12-hl940 Configuration
Miscellaneous filesystems
Arrow keys navigate the menu. <Enter> selects submenus --->. Highlighted letters are
hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc>
to exit, <?> for Help, </> for Search. Legend: [*] built-in [ ] excluded <M> module
< > module capable
< > Amiga FFS file system support (EXPERIMENTAL)
< > Apple Macintosh file system support (EXPERIMENTAL)
< > Apple Extended HFS file system support
< > BeOS file system (BeFS) support (read only) (EXPERIMENTAL)
< > BFS file system support (EXPERIMENTAL)
< > EFS file system support (read only) (EXPERIMENTAL)
<*> YAFFS2 file system support
--- 512 byte / page devices
[*] Lets Yaffs do its own ECC
[ ] Use the same ecc byte order as Steven Hill's nand_ecc.c
--- 2048 byte (or larger) / page devices
[*] Autoselect yaffs2 format
<Select> < Exit > < Help >
Open
1 mdl
Connected to 192.168.0.230:22.
SSH2 xterm 100x26 1 session
数字

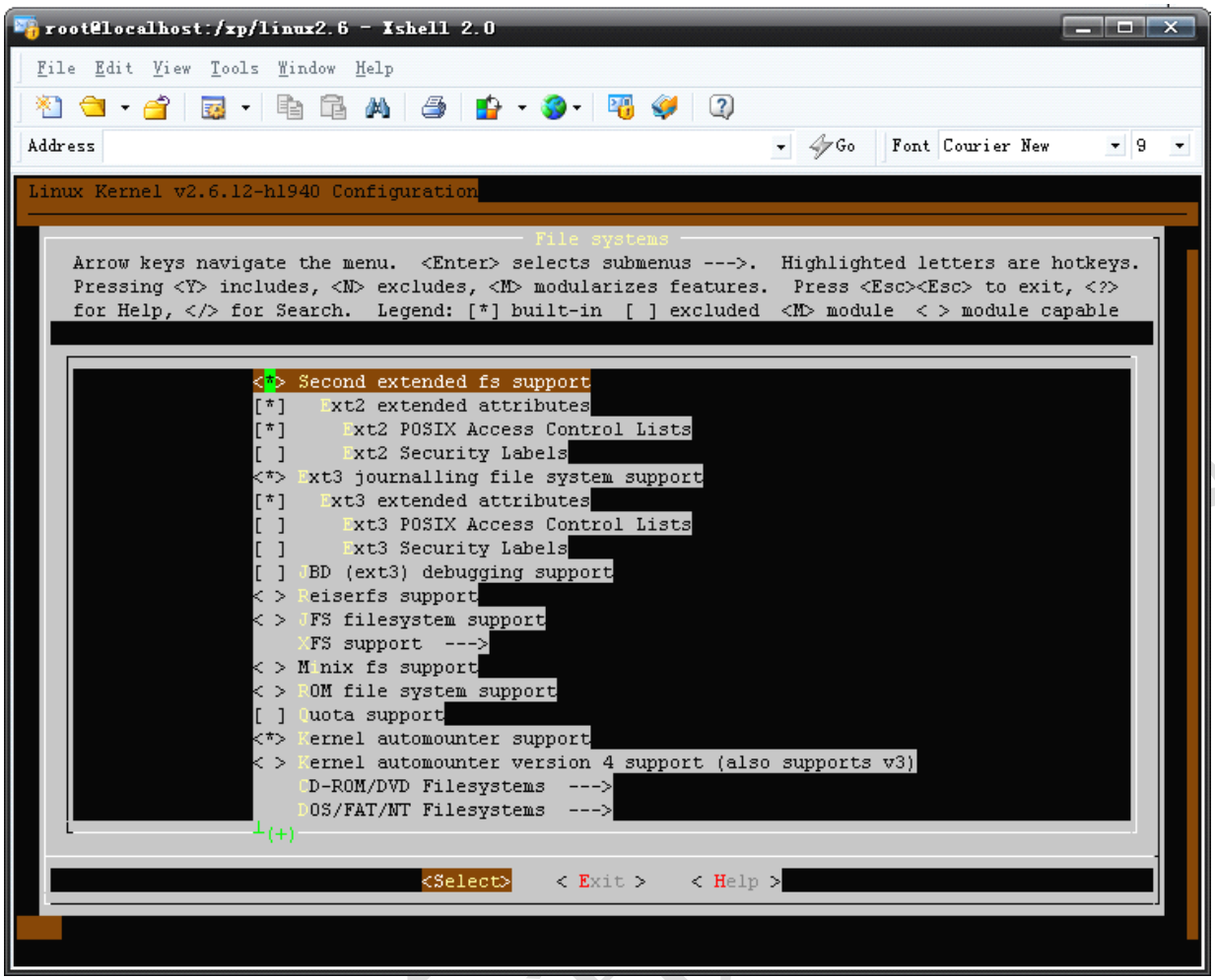
保存退出即可。

2.5.12 配置文件系统（EXT2/EXT3/FAT/NFS）

选择并进入 File systems 菜单

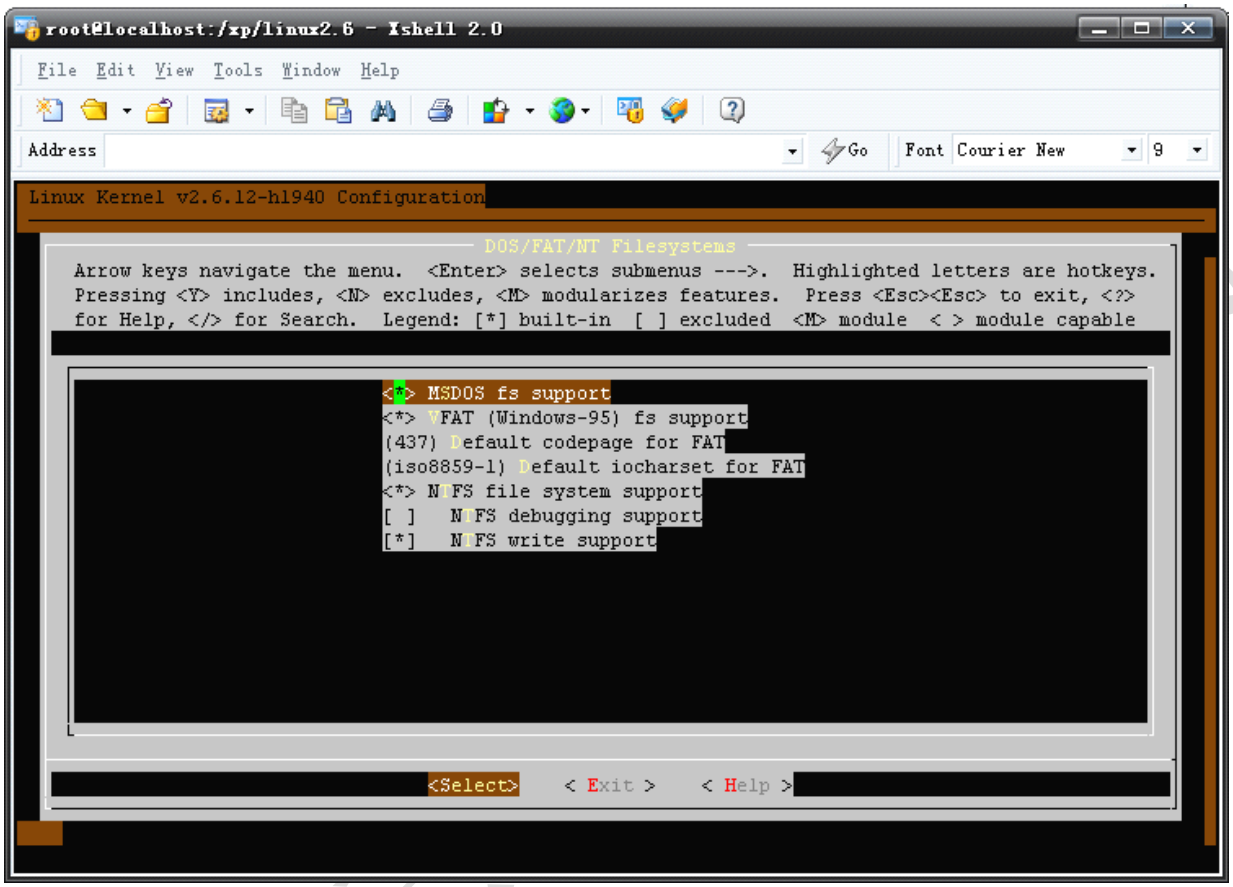


选择如图所示选项，以支持 EXT2/EXT3 文件系统。


root@localhost:/xp/linux2.6 - Xshell 2.0
File Edit View Tools Window Help
Address
Go
Font Courier New
9
Linux Kernel v2.6.12-hl940 Configuration
File systems
Arrow keys navigate the menu. <Enter> selects submenus --->. Highlighted letters are hotkeys.
Pressing <Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?>
for Help, </> for Search. Legend: [*] built-in [ ] excluded <M> module < > module capable
<M> Second extended fs support
[*] Ext2 extended attributes
[*] Ext2 POSIX Access Control Lists
[ ] Ext2 Security Labels
<*> Ext3 journalling file system support
[*] Ext3 extended attributes
[ ] Ext3 POSIX Access Control Lists
[ ] Ext3 Security Labels
[ ] JBD (ext3) debugging support
< > Reiserfs support
< > JFS filesystem support
XFS support --->
< > Minix fs support
< > ROM file system support
[ ] Quota support
<*> Kernel automounter support
< > Kernel automounter version 4 support (also supports v3)
CD-ROM/DVD Filesystems --->
DOS/FAT/NT Filesystems --->
<Select> < Exit > < Help >

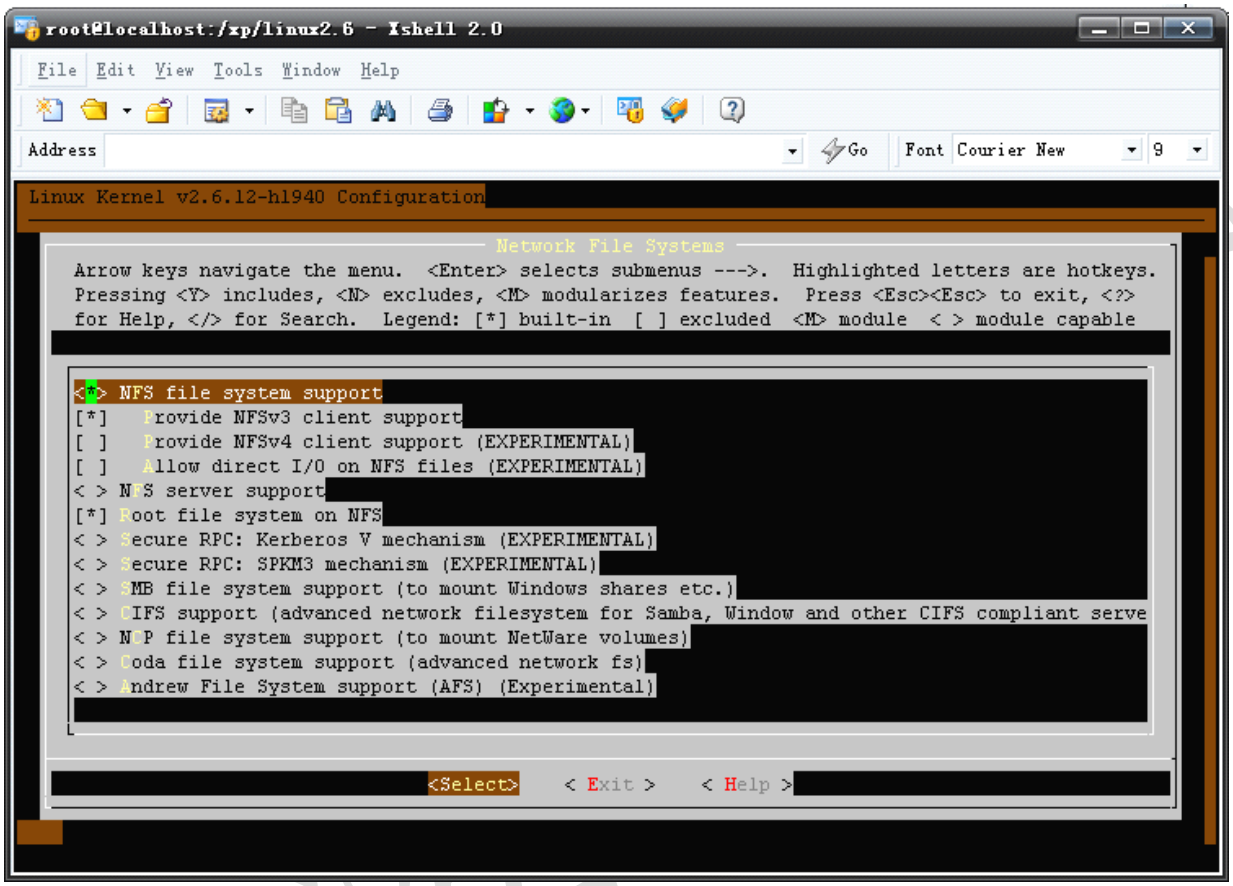
选择并进入 DOS/FAT/NT Filesystems --->

选择如图所示选项。


root@localhost:/xp/linux2.6 - Xshell 2.0
File Edit View Tools Window Help
Address
Go
Font Courier New
9
Linux Kernel v2.6.12-hl940 Configuration
DOS/FAT/NT Filesystems
Arrow keys navigate the menu. <Enter> selects submenus --->. Highlighted letters are hotkeys.
Pressing <Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?>
for Help, </> for Search. Legend: [*] built-in [ ] excluded <M> module < > module capable
<*> MSDOS fs support
<*> VFAT (Windows-95) fs support
(437) Default codepage for FAT
(iso8859-1) Default iocharset for FAT
<*> NTFS file system support
[ ] NTFS debugging support
[*] NTFS write support
<Select> < Exit > < Help >

返回 File systems 菜单，选择并进入 Network File Systems --->

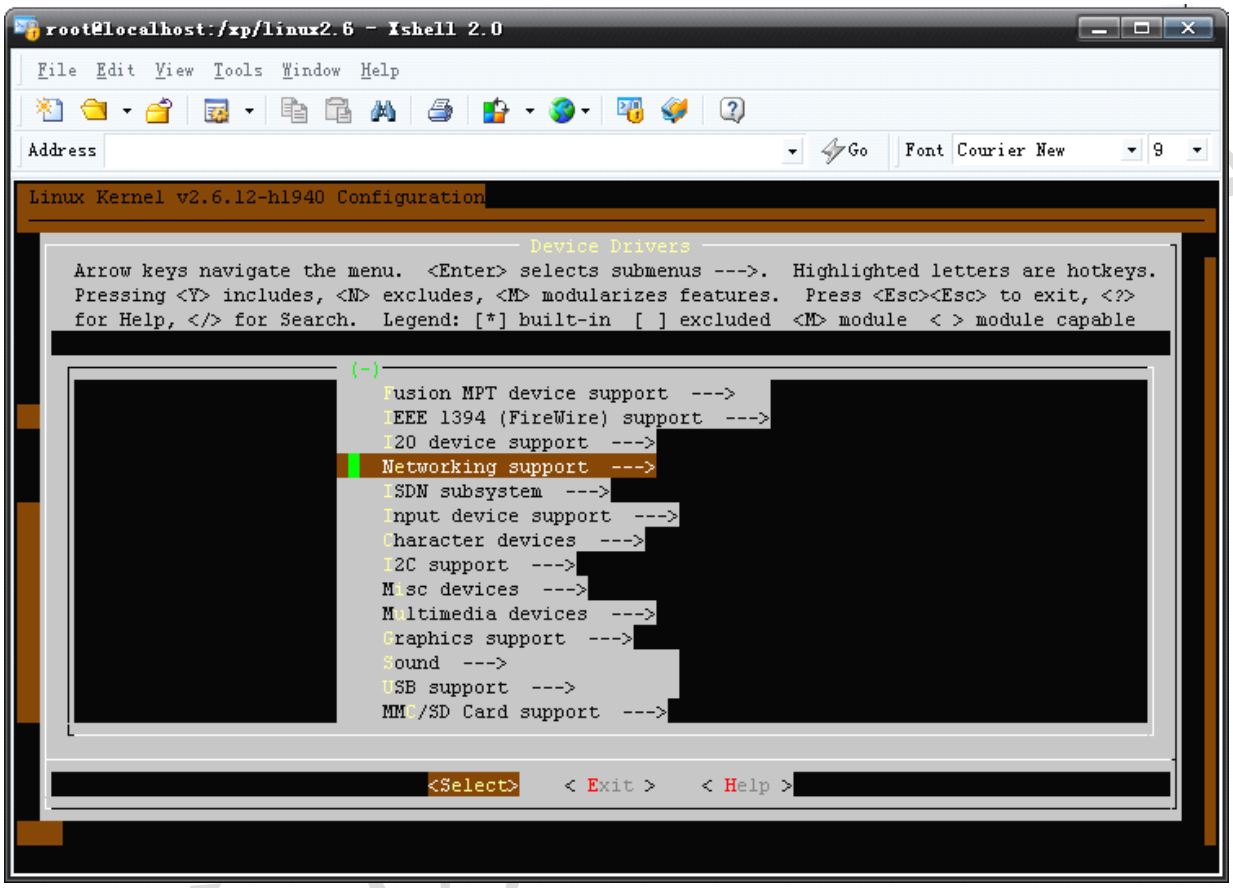
选择如图所示选项以支持 NFS 网络文件系统。


root@localhost:/xp/linux2.6 - Xshell 2.0
File Edit View Tools Window Help
Address
Go
Font Courier New
9
Linux Kernel v2.6.12-hl940 Configuration
Network File Systems
Arrow keys navigate the menu. <Enter> selects submenus --->. Highlighted letters are hotkeys.
Pressing <Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?>
for Help, </> for Search. Legend: [*] built-in [ ] excluded <M> module < > module capable
<*> NFS file system support
[*] Provide NFSv3 client support
[ ] Provide NFSv4 client support (EXPERIMENTAL)
[ ] Allow direct I/O on NFS files (EXPERIMENTAL)
< > NFS server support
[*] Root file system on NFS
< > Secure RPC: Kerberos V mechanism (EXPERIMENTAL)
< > Secure RPC: SPKM3 mechanism (EXPERIMENTAL)
< > SMB file system support (to mount Windows shares etc.)
< > CIFS support (advanced network filesystem for Samba, Window and other CIFS compliant serve
< > NCP file system support (to mount NetWare volumes)
< > Coda file system support (advanced network fs)
< > Andrew File System support (AFS) (Experimental)
<Select> < Exit > < Help >

2.5.13 配置 CS8900 网卡驱动

退回到主菜单，选择进入 Device Drivers 菜单

找到 Networking support，选择进入

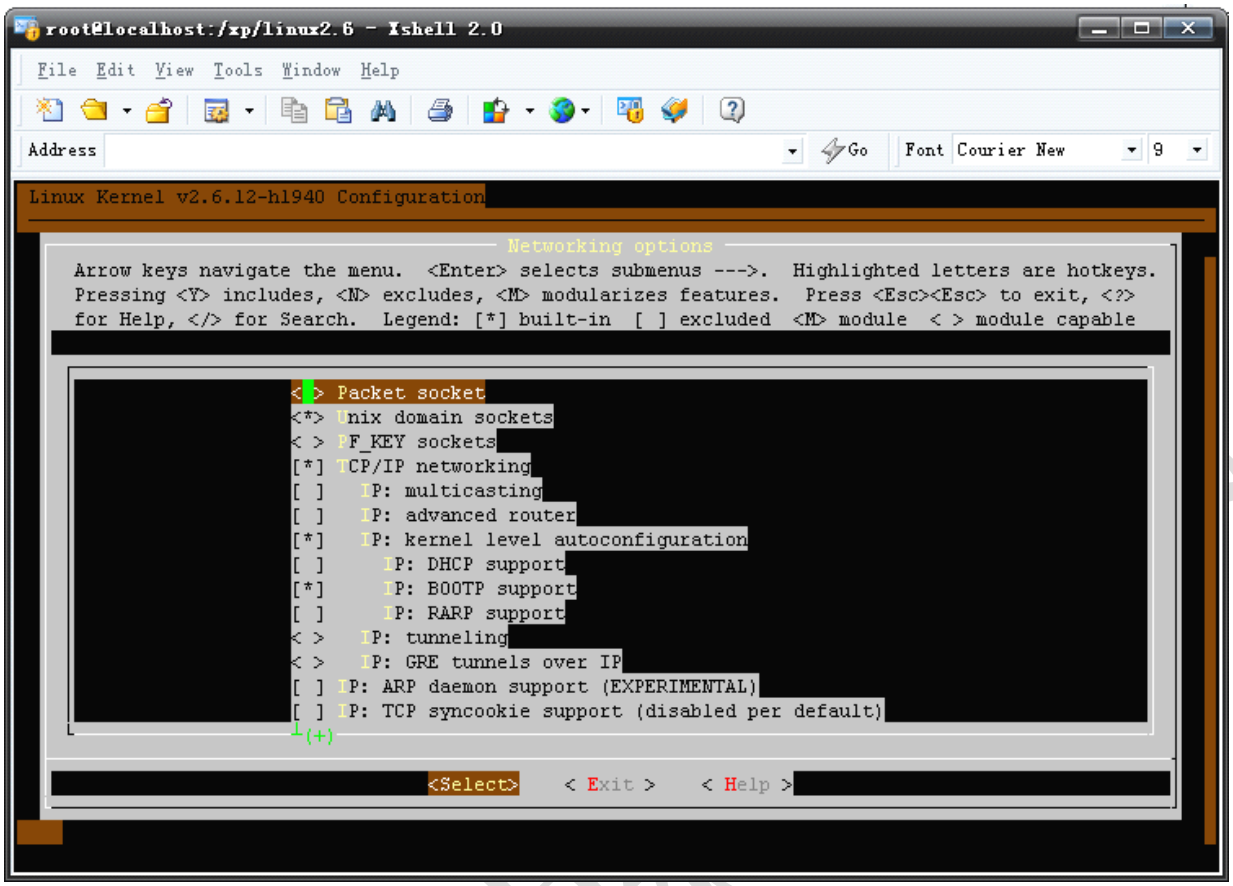

root@localhost:/xp/linux2.6 - Xshell 2.0
File Edit View Tools Window Help
Address
Go
Font Courier New
9
Linux Kernel v2.6.12-hl940 Configuration
Device Drivers
Arrow keys navigate the menu. <Enter> selects submenus --->. Highlighted letters are hotkeys.
Pressing <Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?>
for Help, </> for Search. Legend: [*] built-in [ ] excluded <M> module < > module capable
(-)
Fusion MPT device support --->
IEEE 1394 (FireWire) support --->
I2O device support --->
Networking support --->
ISDN subsystem --->
Input device support --->
Character devices --->
I2C support --->
Misc devices --->
Multimedia devices --->
Graphics support --->
Sound --->
USB support --->
MMC/SD Card support --->
<Select> < Exit > < Help >

首先需要配置网络协议支持，选中

[*] Networking support

选择进入 Networking options ---->

选择如图所示选项

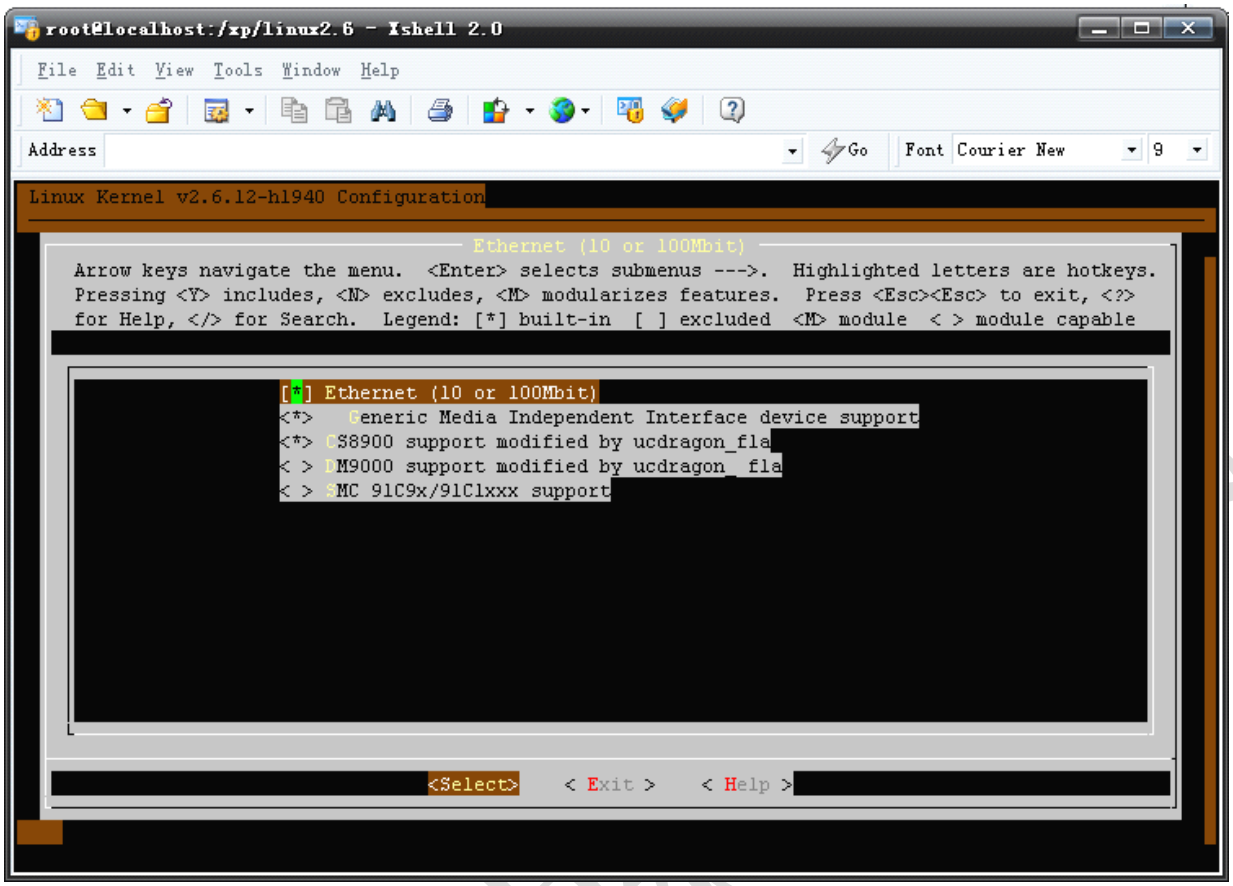

root@localhost:/xp/linux2.6 - Xshell 2.0
File Edit View Tools Window Help
Address
Go
Font Courier New
9
Linux Kernel v2.6.12-hl940 Configuration
Networking options
Arrow keys navigate the menu. <Enter> selects submenus --->. Highlighted letters are hotkeys.
Pressing <Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?>
for Help, </> for Search. Legend: [*] built-in [ ] excluded <M> module < > module capable
< > Packet socket
<*> Unix domain sockets
< > IF_KEY sockets
[*] TCP/IP networking
[ ] IP: multicasting
[ ] IP: advanced router
[*] IP: kernel level autoconfiguration
[ ] IP: DHCP support
[*] IP: BOOTP support
[ ] IP: RARP support
< > IP: tunneling
< > IP: GRE tunnels over IP
[ ] IP: ARP daemon support (EXPERIMENTAL)
[ ] IP: TCP syncookie support (disabled per default)
<Select> < Exit > < Help >

返回上一级目录，选中

[*] Network device support

选择进入 Ethernet (10 or 100Mbit) --->

选择如图所示选项


root@localhost:/xp/linux2.6 - Xshell 2.0
File Edit View Tools Window Help
Address
Go
Font Courier New
9
Linux Kernel v2.6.12-hl940 Configuration
Ethernet (10 or 100Mbit)
Arrow keys navigate the menu. <Enter> selects submenus --->. Highlighted letters are hotkeys.
Pressing <Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?>
for Help, </> for Search. Legend: [*] built-in [ ] excluded <M> module < > module capable
[*] Ethernet (10 or 100Mbit)
<*> Generic Media Independent Interface device support
<*> CS8900 support modified by ucdragon_fla
< > DM9000 support modified by ucdragon_ fla
< > SMC 91C9x/91C1xxx support
<Select> < Exit > < Help >

2.6 cramfs 文件系统制作

1. 将[V1.0]qte.cramfs 拷贝到任意目录下
2. 在该目录下建立两个文件：

   mkdir roms

   mkdir tmp
3. 将 [V1.0]qte.cramfs 挂接到 roms 目录

   mount -t cramfs 2440_mouse.cramfs roms -o loop
4. 将 roms 目录下的内容压缩

   tar -cvf 1.tar roms

   这样将在 roms 的上一级目录产生一个 1.tar 的包
5. 将包解压到 tmp 目录下。

   umount roms ；卸载挂接

```
cd ..    ；进入上一级目录
mv  1.tar   tmp ；
cd  tmp ；
tar  -xvf       1.tar    ；将打包的根文件系统的里的内容解压
rm  1.tar
```

6. 经过上面的步骤就可以将自己的驱动和应用程序添加到 cramfs 根文件系统中了。现在将开始制作 cramfs 根文件系统，你可以进入 tmp \ roms 文件夹，在里面新建文件夹或者把自己做好的驱动或应用程序放进去。
7. 接下来在这个目录下运行命令：

```
mkcramfs  tmp  [V1.0]qte.cramfs
```

运行成功后，会在该目录下生成 [V1.0]qte.cramfs 根文件系统。mkcramfs 工具在光盘：“OK2440\实用工具\linux 交叉编译工具链”下。

8. 根文件系统制作完成

2.7 配置 PC 机 linux 的 ftp 服务

在 PC 机命令行输入

redhat-config-services

打开系统服务配置窗口，在左侧一栏找到 vsftpd 服务选项框，并选中它，然后点 File->Save Changes 保存设置。

2.8 配置 PC 机 linux 的 telnet

开发板上电进入 linux 操作系统并进入到命令行，在命令行输入

redhat-config-services

打开系统服务配置窗口，在左侧一栏找到 telnet 服务选项框，并选中它，然后点 File->Save Changes 保存设置，下次开机就启动了 telnet 服务，如图:

2.9 编译应用程序

首先我们在 PC 机上写一个简单的应用程序：Hello World

```
#include <stdio.h>
  int main()
  {
      printf("Hello World\n")
      return 0;
  }
```

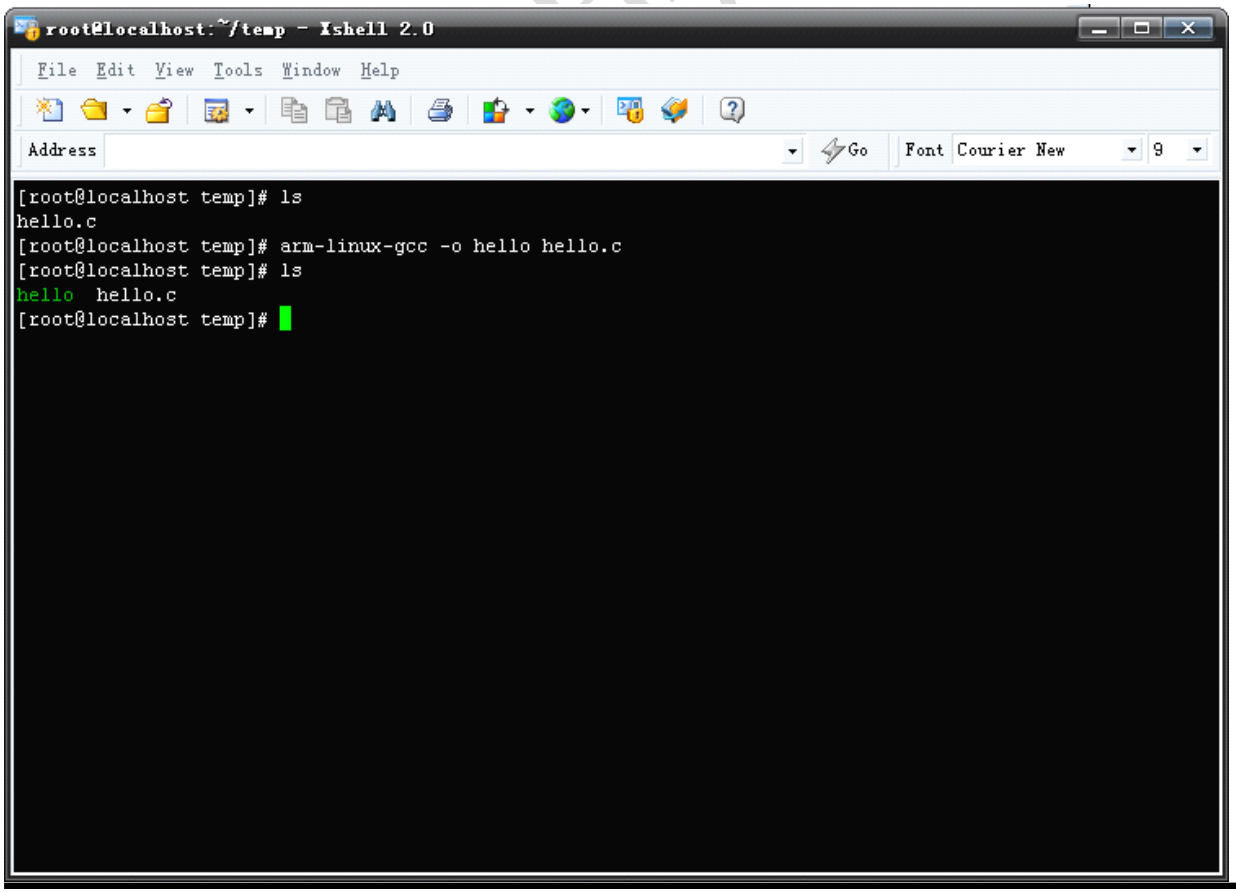
保存为 hello.c

在 Linux 命令行下进入 hello.c 文件所在的目录，输入下面的命令进行编译：

arm-linux-gcc -o hello hello.c

编译成功后可看到新生成的 hello 文件，传到开发板就可以直接运行了。

注：通过交叉编译生成的文件可在 OK2440 开发板上运行，而不能在 PC 机上运行。



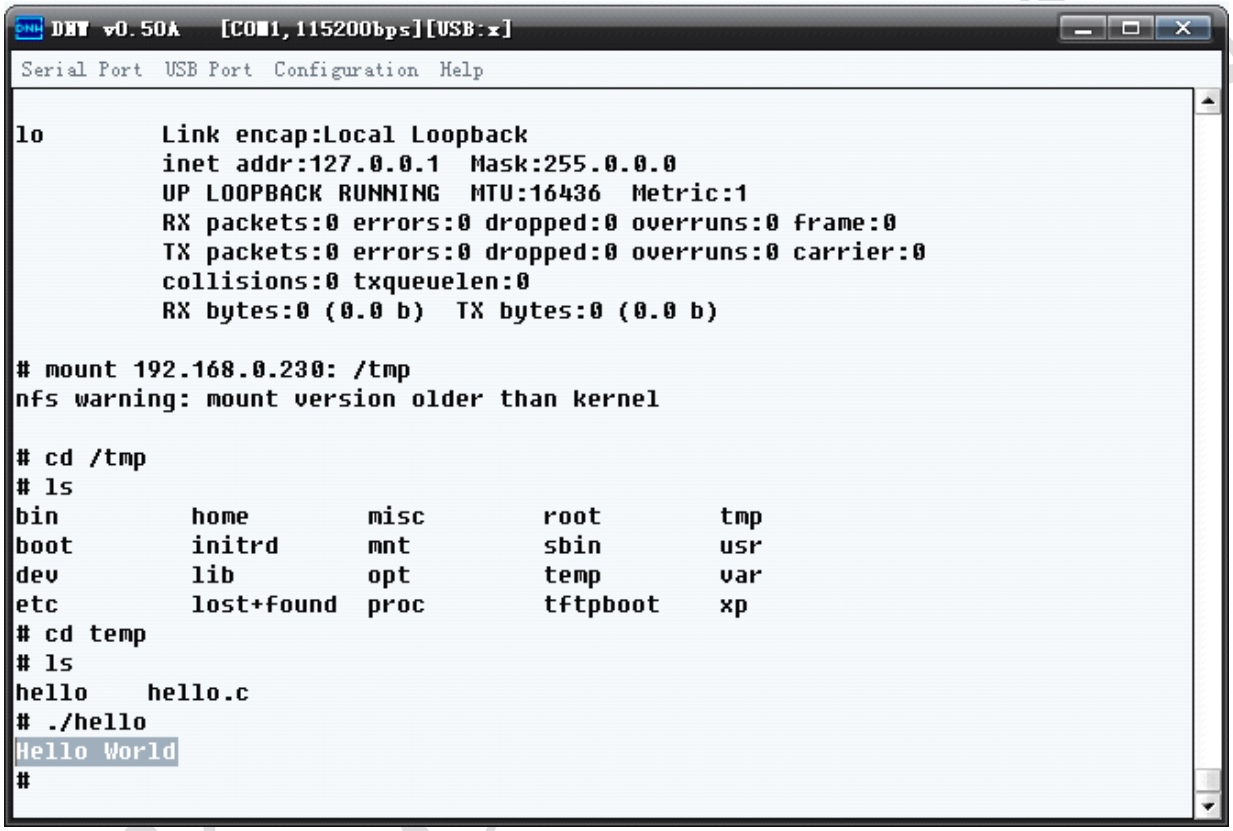
2.10 把应用程序传到目标板上运行

若 PC 端是 Linux 操作系统，可通过挂接网络文件系统，在开发板上执行 PC 机上编译好的程序。参见 1.7 章节。

若 PC 端是 Windows 操作系统，可通过串口传送文件。参见 1.6 章节。

如上一节生成的 hello 文件，传到开发板上后，在该文件所在目录输入下面的命令运行：

```
./hello
```



有时可能会提示没有执行权限，通过以下命令增加可执行权限：

```
chmod +x hello
```

2.11应用程序实验

2.11.1 串口编程

串口是应用中比较重要的一块，下面给出一个简单的串口程序。

serial.c 源码如下：

```
#include "stdio.h"
#include   <sys/ioctl.h>
#include   <termios.h>
#include   <stdlib.h>
#include   <fcntl.h>


struct termios tio;
int fd;
void main()
{
    if((fd = open("/dev/ttyS0",O_RDWR | O_NDELAY |O_NOCTTY)) < 0) //串口实例，失败返回-1
    {
        printf("Couldn't open\n");
        exit(1);
    }
    else
    {
        printf("comm open success! \n");
    }
    tio.c_cflag = B115200 |CS8 |CREAD |CLOCAL; //设置波特率及串口的相关配置
    tio.c_cflag &= ~HUPCL;
    tio.c_lflag = 0;
    tio.c_iflag = IGNPAR;
```

```
    tio.c_oflag = 0;
    tio.c_cc[VTIME] = 0;
    tio.c_cc[VMIN] = 0;                    // 初始化结构体 tio

    tcflush(fd,TCIFLUSH);     //清空输入（输出）缓存
    tcsetattr(fd,TCSANOW,&tio);            //设置终端的相关参数，就是上面配置好的 tio
    fcntl(fd,F_SETFL,FNDELAY);             //该函数可以改变以打开文件的性质
    write(fd,"$R4C10$",7);                 //写 7 个字节的数据
    sleep(1);
    close(fd);
}
```

应用程序写好后就该编译了，此时要用到基础篇中讲过的交叉编译，编译器为 arm-linux-gcc 版本为 3.4.1。如果你已经把该编译器设成默认路径了，则可直接用下边的命令进行编译：

arm-linux-gcc serial.c –o serial

如果还没有设成默认路径，则需指明编译器的路径

/usr/local/arm/3.4.1/bin/arm-linux-gcc serial.c -o serial

最后运行 serial 即可

./serial

2.11.2 网络编程

Socket 接口是 TCP/IP 网络的 API，Socket 接口定义了许多函数或例程，程序员可以用它们来开发 TCP/IP 网络上的应用程序。要学 Internet 上的 TCP/IP 网络编程，必须理解 Socket 接口。

网络的 Socket 数据传输是一种特殊的 I/O，Socket 也是一种文件描述符。Socket 也具有一个类似于打开文件的函数调用 Socket()，该函数返 回一个整型的 Socket 描述符，随后的连接建立、数据传输等操作都是通过该 Socket 实现的。常用的 Socket 类型有两种：流式 Socket （SOCK_STREAM）和数据报式 Socket（SOCK_DGRAM）。流式是一种面向连接的 Socket，针对于面向连接的 TCP 服务应用；数据 报式 Socket 是一种无连接的 Socket，对应于无连接的 UDP 服务应用。

本开板提供的是比较经典的服务器、客户端的 Socket 网络编程实验

Service.c 源码如下：

```
#include   <sys/types.h>
#include   <sys/socket.h>
#include   <stdio.h>
#include   <netinet/in.h>
#include   <arpa/inet.h>
#include   <unistd.h>
#include   <string.h>
#include   <netdb.h>
#include   <sys/ioctl.h>
#include   <termios.h>
#include   <stdlib.h>
#include   <sys/stat.h>
#include   <fcntl.h>
#include   <signal.h>
#include   <sys/time.h>
void main()
{
      pid_t fd;
      int listensock,connsock;
      char recvbuff[100]; //定义要接收的数据缓冲区
      struct sockaddr_in serveraddr; //定义网络套接字地址结构
      listensock = socket(AF_INET,SOCK_STREAM,0); //创建一个套接字，用于监听
      bzero(&serveraddr,sizeof(struct sockaddr)); //地址结构清零
      serveraddr.sin_family = AF_INET; //指定使用的通讯协议族
      serveraddr.sin_addr.s_addr = htonl(INADDR_ANY); //指定接受任何连接
      serveraddr.sin_port = htons(5000); //指定监听的端口
      bind(listensock,(struct sockaddr *)&serveraddr,sizeof(struct sockaddr_in));//给套接口邦定地址
      listen(listensock,1024); //开始监听
      connsock = accept(listensock,(struct sockaddr *)NULL, NULL);
```

```
    //建立通讯的套接字，accept 函数，等待客户端程序使用 connect 函数的连接
    recv(connsock,recvbuff,sizeof(recvbuff),0); //接收服务器的数据
    printf("%s\n",recvbuff); //打印接收到的数据
    sleep(2);
    close(connsock); //关闭通讯套接字
    close(listensock); //关闭监听套接字
}
```

Client.c 源码如下：（头文件与 service.c 同，再次不再重复）

```
int main(int argc,char **argv)
{
    pid_t fd;
    const char buff[] = "Hello! Welcome here    ggggggggggg!\r\n"; //定义要发送的数据缓冲区；
    int sockfd,connsock; //定义一个 socket 套接字，用于通讯
    struct sockaddr_in serveraddr;//定义网络套接字地址结构
    if(argc!= 2)
    {
        printf("Usage: echo ip 地址");
        exit(0);
    }
    sockfd =socket(AF_INET,SOCK_STREAM,0); //创建一个套接字
    bzero(&serveraddr,sizeof(serveraddr));
    serveraddr.sin_family = AF_INET; //指定使用的通讯协议族
    serveraddr.sin_port = htons(5000);//指定要连接的服务器的端口
    inet_pton(AF_INET, argv[1], &serveraddr.sin_addr);
    connect(sockfd,(struct sockaddr *)&serveraddr,sizeof(serveraddr)); //连接服务器
    send(sockfd,buff,sizeof(buff), 0);//向客户端发送数据
    close(sockfd); //关闭套接字
    return(0);
}
```

编译：

gcc service.c –o service

/usr/local/arm/3.4.1/bin/arm-linux-gcc client.c –o client

通过编译方式可以看出 service.c 在 PC 机下运行，而 client.c 在开发板上运行。

运行：先运行服务器 ./service

在运行客户端 ./client 192.168.0.230 (服务器的 IP)

此时在服务器上就可以接受到客户端发过来的数据“Hello! Welcome here gggggggggggg!”

2.11.3 线程编程

下面给出的是一个简单的关于线程的程序，大家要注意线程程序的编写及编译

thread.c 源码如下：

```
#include<stdio.h>
#include<stddef.h>
#include<unistd.h>
#include<pthread.h>
thread1()
{
    printf("Iam the thread\n");
    sleep(1);
}
main()
{
    int ret;
    pthread_t id;
    ret=pthread_create(&id,NULL,(void*)thread1,NULL);                //创建线程
        if(ret!=0)
        {
            printf("thread_send error\n");
```

```
            exit(1);
        }
        while(1)
        {
        printf("I am the main-thread\n");
        sleep(1);
        }
}
```

线程编程就是这种模式，你可以创建多个。

编译的时候就要注意了 # arm-linux-gcc thread.c –o thread –lpthread (线程编译需要加上红色部分 -lpthread)。

2.12 驱动程序实验

系统调用是操作系统内核和应用程序之间的接口，设备驱动程序是操作系统内核和机器硬件之间的接口.设备驱动程序为应用程序屏蔽了硬件的细节，这样在应用程序看来，硬件设备只是一个设备文件， 应用程序可以象操作普通文件一样对硬件设备进行操作.设备驱动程序是内核的一部分，它完成以下的功能:

1.对设备初始化和释放.

2.把数据从内核传送到硬件和从硬件读取数据.

3.读取应用程序传送给设备文件的数据和回送应用程序请求的数据.

4.检测和处理设备出现的错误.

在 Linux 操作系统下有两类主要的设备文件类型，一种是字符设备，另一种是块设备.字符设备和块设备的主要区别是:在对字符设备发出读/写请求时，实际的硬件 I/O 一般就紧接着发生了，块设备则不然，它利用一块系统内存作缓冲区，当用户进程对设备请求能满足用户的要求，就返回请求的数据，如果不能，就调用请求函数来进行实际的 I/O 操作.块设备是主要针对磁盘等慢速设备设计的，以免耗费过多的 CPU 时间来等待.

本篇提供的是关于字符设备的驱动实验。

2.12.1 LED 驱动实验

LED 是由 CPU 的 GPIO（通用可编程 IO 口）控制的。GPIO 一般由两组寄存器控制，即一组控制寄存器和一组数据寄存器。控制寄存器可设置 GPIO 口的工作方式为输入或输出。当引脚为输出时，向数据寄存器的对应位写入 1 和 0 会分别在引脚上产生高低电平，对 led，bell 的控制就是这样的；当引脚设置为输入时，读取数据寄存器的对应位可获得引脚上响应电平信号，按键操作就是这样，做 LED 驱动程序主要是做对 led 对应的 GPIO 口进行操作的函数，还包括定义的设备操作的结构体，驱动编写比较简单，下面是 LED 驱动程序 s3c2440_leds.c 源码：

```
#include <linux/config.h>
#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/fs.h>
#include <linux/init.h>
#include <linux/devfs_fs_kernel.h>
#include <linux/miscdevice.h>
#include <linux/delay.h>
#include <asm/irq.h>
#include <asm/arch/regs-gpio.h>
#include <asm/hardware.h>

#define DEVICE_NAME     "leds"                  //定义设备名
#define LED_MAJOR 231                           //手动定义主设备号

//定义要操作的设备，把每一个 led 作为结构体的一个成员
static unsigned long led_table [] = {
        S3C2410_GPF3,
        S3C2410_GPF4,
        S3C2410_GPF5,
        S3C2410_GPF6,
```

```
    S3C2410_GPF7,

};

//对设备进行设置，结构体的每一个成员是对对应 led 的设置
static unsigned int led_cfg_table [] = {
    S3C2410_GPF3_OUTP,
    S3C2410_GPF4_OUTP,
    S3C2410_GPF5_OUTP,
    S3C2410_GPF6_OUTP,
    S3C2410_GPF7_OUTP,

};

//设备驱动程序中对设备的 I/O 通道进行管理的函数，用来实现对 led 的操作
static int s3c2440_leds_ioctl(
    struct inode *inode,
    struct file *file,
    unsigned int cmd,
    unsigned long arg)
{
    switch(cmd) {
    case 0:
    case 1:
        if (arg > 4) {
            return -EINVAL;
        }
        s3c2410_gpio_setpin(led_table[arg], !cmd);
        return 0;
```

```
    default:
        return -EINVAL;
    }
}
static struct file_operations s3c2440_leds_fops = {
    .owner    =    THIS_MODULE,
    .ioctl    =    s3c2440_leds_ioctl,
};

//模块加载函数
static int __init s3c2440_leds_init(void)
{
    int ret;
    int i;
    //注册设备号
    ret = register_chrdev(LED_MAJOR, DEVICE_NAME, &s3c2440_leds_fops);
    if (ret < 0) {
      printk(DEVICE_NAME " can't register major number\n");
      return ret;
    }

    devfs_mk_cdev(MKDEV(LED_MAJOR,  0),  S_IFCHR  |  S_IRUSR  |  S_IWUSR  |  S_IRGRP,
DEVICE_NAME);

    for (i = 0; i < 5; i++) {
        s3c2410_gpio_cfgpin(led_table[i], led_cfg_table[i]);
        s3c2410_gpio_setpin(led_table[i], 1);
    }
```

```
    printk(DEVICE_NAME " initialized\n");
    return 0;
}

//模块卸载函数
static void __exit s3c2440_leds_exit(void)
{
    devfs_remove(DEVICE_NAME);
    unregister_chrdev(LED_MAJOR, DEVICE_NAME);
}

module_init(s3c2440_leds_init);
module_exit(s3c2440_leds_exit);
}
```

把驱动加载进内核，采用静态加载方式：

1.s3c2440.c 是 led 驱动　放在内核/dreviers/char 目录下　修改该目录下的 Makefile 和 Kconfig

在 Makefile 里面加入

```
obj-$(CONFIG_SBC2440_LEDS) += sbc2440_leds.o
```

在 Kconfig 里面加入

```
config SBC2440_LEDS
  tristate "SBC2440 LEDs Driver"
  depends on ARCH_S3C2410
  help
    SBC2440 User leds.
```

配置内核的方法见 4.5.9 节。

该驱动的测试程序 ledshow.c 代码如下：

```
#include <stdio.h>
#include <stdlib.h>
```

```
#include <unistd.h>
#include "stdio.h"
#include "sys/types.h"
#include "sys/ioctl.h"
#include "stdlib.h"
#include "termios.h"
#include "sys/stat.h"
#include "fcntl.h"
#include "sys/time.h"

int main()
{
    int on=1;
    int led;
    int fd;
    fd = open("/dev/leds", 0);
    if (fd < 0) {
                perror("open device leds");
                exit(1);
                }
    printf("leds test show. press ctrl+c to exit    \n");
    while(1)
        {
          for(led=0;led<5;led++)
            {
                ioctl(fd, on, led);
                usleep(50000);
            }
          on=!on;
```

```
            }
        close(fd);
        return 0;
}
```

用交叉编译器编译器编译 ledshow.c　生成二进制可执行代码，下载到开发板执行，实现流水的效果

2.12.2 AD 驱动实验

开发板为客户提供 2440 八通道 AD 中的第 0 通道的 AD 转换接口，其驱动程序 adc2440.c 源码如下：

```
#include <linux/kernel.h>
#include <linux/module.h>

#include <asm/uaccess.h>
#include <linux/mm.h>
#include <linux/ioport.h>

#include <asm/io.h>
#include <linux/ioport.h>
#include <linux/fs.h>



#define ADC_MAJOR 102                   //定义主设备号
#define DEVICE_NAME    "adc_dev"        //定义设备名称
#define SUCCESS      0

#define adc_con           (unsigned long)ioremap(0x58000000,4)
```

```
#define adc_dat0          (volatile unsigned long)ioremap(0x5800000c,4)


static int Device_Open = 0;


int adc_init(void);
void adc_cleanup(void);
static int device_open(struct inode *,struct file *);
static int device_release(struct inode *,struct file *);
static ssize_t device_read(struct file *,char *,size_t,loff_t *);
int init_module(void);
void cleanup_module(void);



struct file_operations    adc_ops =                  // 该结构包含在<linux/fs.h>中
{                                                    //结构中的每个字段指向驱动程序中实现特定的函数
      .owner    =    THIS_MODULE,
      .read     =    device_read,
      .open     =    device_open,
      .release  =    device_release,
};
static int Major;
int __init adc_init(void)                    //初始化
{

      Major = register_chrdev(ADC_MAJOR,DEVICE_NAME,&adc_ops);   //注册操作，返回主设备值
      if(Major <0)
      {
            printk("ADC init_module:failed with %d\n",Major);       //主设备号小于 0，则注册失败
            return Major;
```

```
    }

    devfs_mk_cdev(MKDEV(ADC_MAJOR, 0), S_IFCHR | S_IRUSR | S_IWUSR | S_IRGRP,
DEVICE_NAME);

    printk(DEVICE_NAME " initialized\n");

    return 0;
}

static int device_open(struct inode * inode,struct file *file)          //与应用程序中 open 函数对应的驱动函数
{
    if(Device_Open)
    {
        return -EBUSY;
    }
    Device_Open++;
    return SUCCESS;
}

static int device_release(struct inode * inode,struct file *file)
{
    Device_Open --;
    return 0;
}

static ssize_t device_read(struct file *file,
                           char * buffer,
                           size_t length,
```

```
                                    loff_t * offset)
{
    unsigned long buf;
    __raw_writel((1<<14)|(0x5<<6), adc_con);          //设置 ADCCON
    __raw_writel(__raw_readl(adc_con)|0x1, adc_con);   //启动 AD 转换
    while(__raw_readl(adc_con) &0x1);                  //启动转换后，等待启动位清零
    while(!(__raw_readl(adc_con) & 0x8000));               //等待转换是否完毕
    buf=__raw_readl(adc_dat0) & 0x3ff;                     //取出转换后得到的有效数据
    *buffer=(unsigned char)buf;
    buffer++;
    *buffer=(unsigned char)(buf>>8);
    return 2;
}
void adc_cleanup()                  //卸载驱动时执行该函数
{
    int ret;
    ret = unregister_chrdev(Major,DEVICE_NAME);       //注销驱动
    if(ret < 0)                                        //返回值小于 0 时注销失败
    {
        printk("unregister_chrdev:error %d\n",ret);
    }
}
module_init(adc_init);
module_exit(adc_cleanup);
MODULE_LICENSE("GPL");
```

该程序的 makefile 文件代码如下：

```
obj-m := adc2440.o
KERNELDIR ?= /xp/linux-2.6.12                  //内核路径
```

```
PWD := $(shell pwd)



default:
	$(MAKE) -C $(KERNELDIR) M=$(PWD) modules
```

有了 adc2440.c 和 makefile 两个文件后运行名 make 命令即可的到 adc2440.ko 文件

为测试该驱动程序，我们提供 AD 的应用程序 adc.c 源码如下：

```
#include "stdio.h"
#include "sys/types.h"
#include "sys/ioctl.h"
#include "stdlib.h"
#include "termios.h"
#include "sys/stat.h"
#include "fcntl.h"
#include "sys/time.h"
main()
{
        int fd,ret，adcdat0;
        char buf[2];
        fd=open("/dev/adc_dev",O_RD);
        if(fd<0)
        {
                printf("open adc error\n");
                exit(1);
        }
        else
        {
                printf("success\n");
                while(1)
```

```
                {
                        read(fd,buf,2);
                        adcdat0=buf[1];
                        adcdat0=adcdat0<<8||buf[0]
                        printf("%x \n",adcdat0);
                        sleep(1);
                }
        }
        close(fd);
}
```

交叉编译后得到 adc 文件。

在测试时需先手动加载驱动然后运行执行应用程序，在不需要的时候可以卸载驱动。命令如下：

insmod adc2440.ko　　//加载驱动

./adc　　//运行应用程序

rmmod adc2440　　//卸载驱动

三.Linux 图形编程

3.1 用户界面 GUI 介绍

所谓 GUI(Graphics User Interface)，就是图形用户界面。图形用户界面的广泛流行是当今计算机技术的重大成就之一，它极大地方便了非专业用户的使用，人们不再需要死记硬背大量的命令，而可以通过窗口、菜单方便地操作。它的主要特征有三点：

WIMP。其中，W（Windows）指窗口，是用户或系统的一个工作区域。一个屏幕上可以有多个窗口。I（Icons）指图图标，是形象化的图形标志，易于人们隐喻和理解。M（Menu）指菜单，可供用户选择的功能提示。P（Pointing Devices）指鼠 标等，便于用户直接对屏幕对象进行操作。

用户模型。GUI 采用了不少 Desktop 桌面办公的隐喻，让使用者共享一个直观的界面框架。由于人们熟悉办公桌的情况，因而对计算机显示的图标的含义容易理解，诸如：文件夹、收件箱、画笔、工作簿、钥匙及时钟等。

直接操作。过去的界面不仅需要记忆大量命令，而且需要指定操作对象的位置， 如行号、空格数、X 及 Y 的坐标等。采用 GUI 后，用户可直接对屏幕上的对象进 行操作，如拖动、删除、插入以及放大和旋转等。用户执行操作后，屏幕能立即给出反馈信息或结果 ，称为所见即所得（ What You See Is What You Get， WYSIWYG）。用视、点（鼠标）代替了记、击（键盘），给用户带来了方便。

通常所见的 GUI 都是位于 PC 机上的，但是在 PC 上 GUI 并不适合嵌入式系统。嵌入式设备有严格的资源要求（比如十分有限的存储空间）。同时嵌入式系统经常有一些特殊的 要求，而普通的 PC 上的图形窗口系统是不能满足这些要求的。比如特殊的外观效果，要提 供给用户的函数，提高装载速度，特殊的底层图形或输入设备。由此可见嵌入式系统必定要 有自己的 GUI。

综上所述，嵌入式 GUI 就是在嵌入式系统中为特定的硬件设备或环境而设计的图形用

户界面系统。所以嵌入式 GUI 不但要具有以上有关 GUI 的特征，而且在实际应用中，嵌入 式系统对它来说还有如下的基本要求：

轻型，占用源少；

高性能；

高可靠性；

可配置性。

下面我们就目前市场上常用的嵌入式 GUI 做一个简单的介绍。

3.1.1 常用 GUI 介绍

1．MiniGUI

由北京飞漫软件技术有限公司开发的 MiniGUI(http://www.minigui.org),是国内为数不多 的几大国际知名自由软件之一。MiniGUI 是面向实时嵌入式系统的轻量级图形用户界面支 持系统，1999 年初遵循 GPL 条款发布第一个版本以来，已广泛应用于手持信息终端、机顶盒、工业控制系统及工业仪表、彩票机、金融终端等产品和领域。目前，MiniGUI 已成为跨操作系统的图形用户界面支持系统，可在 Linux/uClinux、eCos 、 uC/OS-II 、 VxWorks 等操作系统上运行；已验证的硬件平台包括 Intelx86 、ARM （ ARM7/AMR9/StrongARM/xScale）、PowerPC、MIPS、M68K（DragonBall/ColdFire）等等。

MiniGUI 良好的体系结构及优化的图形接口，可确保最快的图形绘制速度。在设计之初，就充分考虑到了实时嵌入式系统的特点，针对多窗口环境下的图形绘制开展了大量的研究及开发，优化了 MiniGUI 的图形绘制性能及资源占有。MiniGUI 在大量实际系统中的应用，尤其在工业控制系统的应用，证明 MiniGUI 具有非常好的性能。

2．MicroWindows

MicroWindows（http://microwindows.censoft.com）是一个开放源码的项目，目前由美国 Century Software 公司主持开发。该项目的开发一度非常活跃，国内也有人参与了其中的开 发，并编写了 GB2312 等字符集的支持。但在 Qt/Embedded 发布以来，该项目变得不太活跃， 并长时间停留在 0.89Pre7 版本。可以说，以开放源码形势发展的 MicroWindows 项目，基本 停滞。

MicroWindows 是一个基于典型客户/服务器体系结构的 GUI 系统，基本分为三层。最底层是面向图形输出和键盘、鼠标或触摸屏的驱动程序；中间层提供底层硬件的抽象接口，并进行窗口管理；最高层分别提供兼容于 X Window 和 Windows CE（Win32 子集）的 API。 该项目的主要特色在于提供了类似 X 的客户/服务器体系结构，并提供了相对完善的图形功能，包括一些高级的功能，比如 Alpha 混合，三维支持，TrueType 字体支持等。MicroWindows 采用 MPL（mozilla public license）条款发布。

3．DinX

DinX 非常适合于在很小的系统上运行多窗口程序，它简单、轻巧，并且快速。DinX 并 不是 X，它使用 Linux 核心的 framebuffer 视频驱动，采用 Client/Server 模式。为此，系统提 供了两个界面：/dev/dinxsvr 和/dev/dinxwin。

一个服务器程序连接到/dev/dinxsvr，并决定来自各程序窗口的 request 各占有视屏的各个部分。它也负责给各窗口发送像鼠标移动这样的事件消息。Clinet 程序连接到/dev/dinxwin，与 Server 进行消息通信等。Server 进程还负责处理事件、窗口管理、调色板配置等功能。 DinX 是一个实验性的窗口系统，它处在发展阶段中，还存在一些缺陷和问题。DinX 的 license 属于 MPL，也可以转化为 GPL。这样，DinX 核心模块可以集成到 Linux 中，DinX 库可以 链接到其他的 GPL 程序中。

4．OpenGUI

OpenGUI（http://www.tutok.sk/fastgl/）在 Linux 系统上存在已经很长时间了。最初的名 字叫 FastGL，只支持 256 色的线性显存模式，但目前也支持其他显示模式，并且支持多种 操作系统平台，比如 MS-DOS、QNX 和 Linux 等等，不过目前只支持 x86 硬件平台。OpenGUI 也分为三层。最低层是由汇编语言编写的快速图形引擎；中间层提供了图形绘制 API，包括 线条、矩形、圆弧等，并且兼容于 Borland 的 BGI API。第三层用 C++编写，提供了完整的 GUI 对象集。OpenGUI 采用 LGPL 条款发布。OpenGUI 比较适合于基于 x86 平台的实时系统，可移植性稍差，目前的发展也基本停滞。

5．Qt/Embedded

Qt/Embedded 是著名的 Qt 库开放商开发的面向嵌入式系统的 Qt 版本。这个版本的主要 特点是可移植性较好，许多基于 Qt 的 X Window 程序可以非常方便地移植到嵌入式版本。 在下面我们将会详细介绍。

3.1.2 关于 Qt

Qt 是一个跨平台的 C++图形用户界面库，由挪威 TrollTech 公司(www.trolltech.com)出品，它的目的是提供开发应用程序用户界面部分所需要的一切，主要通过汇集 C++类的形式来实现这一目的。它提供给应用程序开发者建立艺术级的图形用户界面所需的所用功能。Qt 是完全面向对象的，很容易扩展的，并且允许真正地组件编程的 GUI 开发工具。

Qt 是 Trolltech 公司的一个标志性产品。Trolltech 公司 1994 年成立于挪威，但是公司的核 心开发团队已经在 1992 年开始了 Qt 产品的研发，并于 1995 年推出了 Qt 的第一个商业版，直 到现在 Qt 已经被世界各地的跨平台软件开发人员使用，而 Qt 的功能也得到了不断的完善和提高。

QT/X11 和 QTE（QT Embedded）是它其中的两个版本。Qt/X11 是基于 X Windows 系统的 Qt 版本，KDE 便是基于它来构建的。为了适用于嵌入式系统，该公司将 Qt/X11 进行了裁减， 发布了 QTE（QT Embedded）版本。QTE 直接基于 Linux 中的 FrameBuffer 设备，删除了 Qt/X11 中一些对资源要求很高的类实现。所以，基于 QTE 实现的应用，不作修改重新编译后，就可 以在 Qt/X11 上运行，而反过来便不可以。

QPE（Qt Plamtop Environment）是 Trolltech 公司所推出的针对 PDA 软件的整体解决方案， 包含了从底层的 GUI 系统、Window Manager、Soft Keyboard 到上层的 PIM、浏览器、多媒体 等方面。目前 QPE 的高版本已更名为 Qtopia，其包含了更多功能。

Qt 是一个支持多操作系统平台的应用程序开发框架，它的开发语言是 C++。Qt 最初主要是为跨平台的软件开发者提供统一的，精美的图形用户编程接口，但是现在它也提供了统一的网络和数据库操作的编程接口。正如微软当年为操作系统提供了友好，精致的用户界面一 样，今天由于 Trolltech 的跨平台开发框架 Qt 的出现，也使得 UNIX、LINUX 这些操作系统以更加方便、精美的人机界面走近普通用户。

Qt 是以工具开发包的形式提供给开发者的，这些工具开发包包括了图形设计器，字体国 际化工具，Makefile 制作工具，Qt 的 C++类库等等；谈到 C++的类库我们自然会想到 MFC， 是的，Qt 的类库也是等价于 MFC 的开发库，但是 Qt 的类库是支持跨平台的类库，也就是说 Qt 类库封装了适应不同操作系统的访问细节，这正是 Qt 的魅力所在。

目前 Qt 系列的软件主要包括 Qt，基于 Framebuffer 的 Qt Embedded，快速开发工具 QtDesigner，国际化工具 Qt Linguist 等部分 。

Trolltech 公司在 1994 年成立，但是在 1992 年，成立 Trolltech 公司的那批程序员就已经开始设计 Qt 了，Qt 的第一个商业版本于 1995 年推出然后 Qt 的发展就很快了，下面是 Qt 发展史上的一些里程碑:

1996 Oct KDE 组织成立

1998 Apr 05 Trolltech 的程序员在 5 天之内将 Netscape5.0 从 Motif 移植到 Qt 上

1998 Apr 08 KDE Free Qt 基金会成立

1998 Jul 09 Qt 1.40 发布

1998 Jul 12 KDE 1.0 发布

1999 Mar 04 QPL 1.0 发布

1999 Mar 12 Qt 1.44 发布

1999 Jun 25 Qt 2.0 发布

1999 Sep 13 KDE 1.1.2 发布

2000 Mar 20 嵌入式 Qt 发布

2000 Sep 06 Qt 2.2 发布

2000 Oct 05 Qt 2.2.1 发布

2000 Oct 30 Qt/Embedded 开始使用 GPL 宣言

2000 Sep 04 Qt free edition 开始使用 GPL

基本上，Qt 同 X Window 上的 Motif，Openwin，GTK 等图形界面库和 Windows 平 台上的 MFC，OWL，VCL，ATL 是同类型的东西，但是 Qt 具有下列优点：

优良的跨平台特性: Qt 支持下列操作系统: Microsoft Windows 95/98， Microsoft Windows NT， Linux， Solaris， SunOS， HP-UX， Digital UNIX (OSF/1， Tru64)， Irix， FreeBSD，BSD/OS，SCO，AIX，OS390，QNX 等等；

面向对象：Qt 的良好封装机制使得 Qt 的模块化程度非常高，可重用性较好，对于用户开发来说是非常方便的。Qt 提供了一种称为 signals/slots 的安全类型来替代 callback，这使得各个元件之间的协同工作变得十分简单；

丰富的 API： Qt 包括多达 250 个以上的 C++类，还提供基于模板的 collections， serialization，file，I/O device，directory management，date/time 类。甚至还包括正 则表达式的处理功能；

支持 2D/3D 图形渲染，支持 OpenGL；

大量的开发文档；

XML 支持；

3.1.3 Qt/Embedded 简介

Qt/Embedded 是一个为嵌入式设备上的图形用户接口和应用开发而订做的 C++工具开 发包。它通常可以运行在多种不同的处理器上部署的嵌入式 Linux 操作系统上。如果不考虑 X Window 窗口系统的需要，居于 Qt/Embedded 的应用程序可以直接对缓冲帧进行写操作。除 了类库以外，Qt/Embedded 还包括了几个提高开发速度的工具，使用标准的 Qt API，我们可 以非常熟练的在 Windows 和 Unix 编程环境里开发应用程序。

Qt/Embedded 是一 组 用于 访 问 嵌入式 设 备 的 Qt C++ API ； Qt/Embedded 的 Qt/X11 ，Qt/Windows 和 Qt/Mac 版本提供的都是相同的 API 和工具。Qt/Embedded 还包括类库以及支持 嵌入式开发的工具。

Qt/Embedded 提供了一种类型安全的被称之为信号与插槽的真正的组件化编程机制，这种机制和以前的回调函数有所不同。Qt/Embedded 还提供了一个通用的 widgets 类，这个类可 以很容易的被子类化为客户自己的组件或是对话框。针对一些通用的任务，Qt 还预先为客户 定制了像消息框和向导这样的对话框。

运 行 Qt/Embedded 所 需 的系统 资 源 可以 很 小 ， 相 对 X 窗口 下的嵌入 解 决 方 案 而 言 ， Qt/Embedded 只要求一个较小的存储空间（Flash）和内存。Qt/Embedded 可以运行在不同的 处理器上部署的 Linux 系统，只要这个系统有一个线性地址的缓冲帧并支持 C++的编译器。 你可以选择不编译 Qt/Embedded 某些你不需要的功能，从而大大减小了它的内存占有量。

Qt/Embedded 包括了它自身的窗口系统，并支持多种不同的输入设备。 开发者可以使用他们熟悉的开发环境来编写代码。Qt 的图形设计器(designer)可以用来可视化地设计用户接口，设计器中有一个布局系统，它可以使你设计的窗口和组件自动根据 屏幕空间的大小而改变布局。开发者可以选择一个预定义的视觉风格，或是建立自己独特的 视觉风格。使用 UNIX/LINUX 操作系统的用户，可以在工作站上通过一个虚拟缓冲帧的应用 程序仿真嵌入式系统的显示终端。

Qt/Embedded 也提供了许多特定用途的非图形组件，例如国际化，网络和数据库交互组件。

Qt/Embedded 是成熟可靠的工具开发包，它在世界各地被广泛使用。除了在商业上的许多应用以外，Qt/Embedded 还是为小型设备提供的 Qtopia 应用环境的基础。Qt/Embedded 以简洁的系统，可视化的表单设计和详致的 API 让编写代码变得愉快和舒畅。

3.1.4 Qtopia 介绍

Qtopia 是 Trolltech 为采用嵌入式 Linux 操作系统的消费电子设备而开发的综合应用平台, Qtopia 包含完整的应用层、灵活的用户界面、窗口操作系统、应用程序的启动程序以及开发 框架。

Qtopia 的特性如下表所示：

窗口操作系统	游戏和多媒体	工作辅助应用程序
同步框架	PIM 应用程序	Internet 应用程序
开发环境	输入法	Java 集成
本地化支持	个性化选项	无线支持

Trolltech 提供三大 Qtopia 版本：Qtopia 手机版、Qtopia PDA 版和 Qtopia 消费电子产品

平台：

Qtopia 手机版：Trolltech 面向嵌入式 Linux 的 Qtopia 手机版(Qtopia Phone)是 Qtopia 的一个 自定义版本，用于内存有限的智能手机和功能手机。它的用户界面可自定义内存占用量低， 手机制造商使用它可以创建令人赞叹的图形用户界面，从而令手机卓越超群。Qtopia 手机版 有两个版本的键盘驱动和手写笔驱动。

Qtopia PDA 版：Qtopia PDA 版 Qtopia PDA 是一个强大的平台，专用于基于 Linux 操作系 统的 PDA，个人数字助理设备等。许多 PDA 都已采用了 Qtopia，Qtopia PDA 版已经成了事 实上的 Linux 标准，它代表了可行的第三种 PDA 设计方案--Qtopia PDA 版，具有可定制的用 户界面，支持多种不同的屏幕尺寸，以及横向和纵向布局。

Qtopia 消费电子产品平台：While Qtopia 手机版和 Qtopia PDA 版是针对移动电话和 PDA 制造商的统包解决方案，而 Qtopia 消费电子产品平台(Qtopia CEP) 则是一套高层次开发平 台，适用于那些希望自行设计和开发应用套件的制造商。Qtopia CEP 使得制造商能够在形形 色色的手写笔和键盘驱动的设备上，创建自定义的环境。这些设备包括电视机，Web，Pad， 无线联网板，机顶盒，以及许多其它基于 Linux 的设备等等。

Qtopia 产品系列旨在为基于 Linux 的消费电子设备提供和创建图形用户界面，它为制造 商提供了前所未有的灵活性和众多选择。

推荐访问的 Qt 资源包括以下几种：

- Trolltech 的主页 http://www.trolltech.com/
- 支持匿名访问的 FTP：ftp://www.trolltech.com 在此网站可以下载到各种版本的 QT、QTE、Qtopia 源码。您可以下载源码后自行编译安装。
- 新闻组服务器：nntp.trolltech.com
- 非官方的 Qt 文档中文翻译小组：http://www.qiliang.net/qt/index.html

3.2 Qt 常用工具的介绍

3.2.1 Qt 设计器（Qt Designer）

如果用户有 Windows 下 GUI 编程的经验，一定不会反对使用可视化工具辅程序开发工作。如果不使用某个代码生成工具，而采用手动方式编写大量的对话框，其工作量之大。并 不是初学者才使用此工具，每一个最求效率的程序员都应该习惯于 GUI 设计工具。

Qt Designer 是 Qt 的可视化图形工具。从前面的实验中，可以发现，要在对话框中对齐控件不是一件容易的事情（需要不断的调整 setGeometry()或者 resize()函数）。Qt 提供了一套 管理空间摆放位置的机制，但是使用这些机制对于普通的应用显得有些麻烦。早在 Qt2.0 版 本出现之前，一些计算机爱好者就自己动手编写了利用图形化方法设计对话框的工具，Qt 的开发者也认识到这点，因此从 2.2 版本后提供了一个 Qt Designer 这个图形设计工具。下图 4-1 就是 Qt Designer 的主窗口：

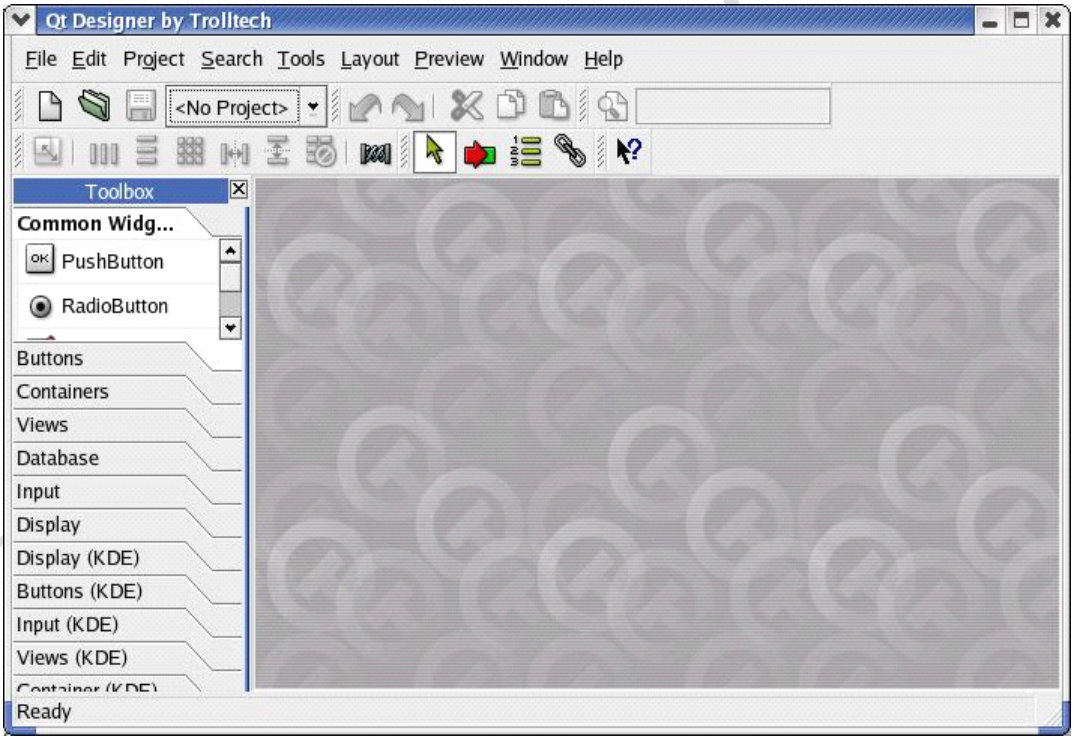


图 4-1

在编译 Qt 的时候会自动编译 Qt Designer。

现在开始创建第一个对话框。从菜单中选择 File|New 命令，或者按快捷键“Ctrl+N”，将看到下面这个对话框，如图 4-2。

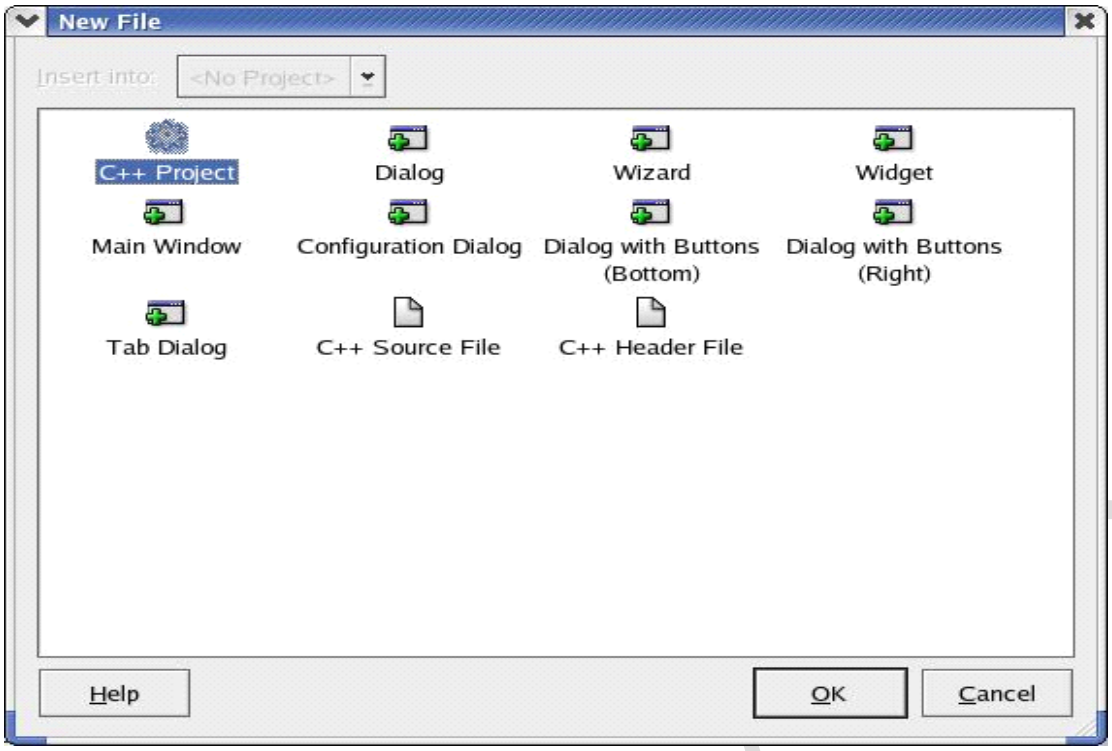


图 4-2

可以在这里选择类型，最常见的可能是 Dialog 或者 Widget。两者的区别在于：Dialog 总是一个单独的窗口，而 Widget 可能被放在其它窗口或者控件中。所有在 Qt Designer 中设计的东西都被称作为表单“Form”，既可以是控件、对话框，也可以是其它类型。在这里选择对话框 Dialog，空的对话框看起来如图 4-3 所示：

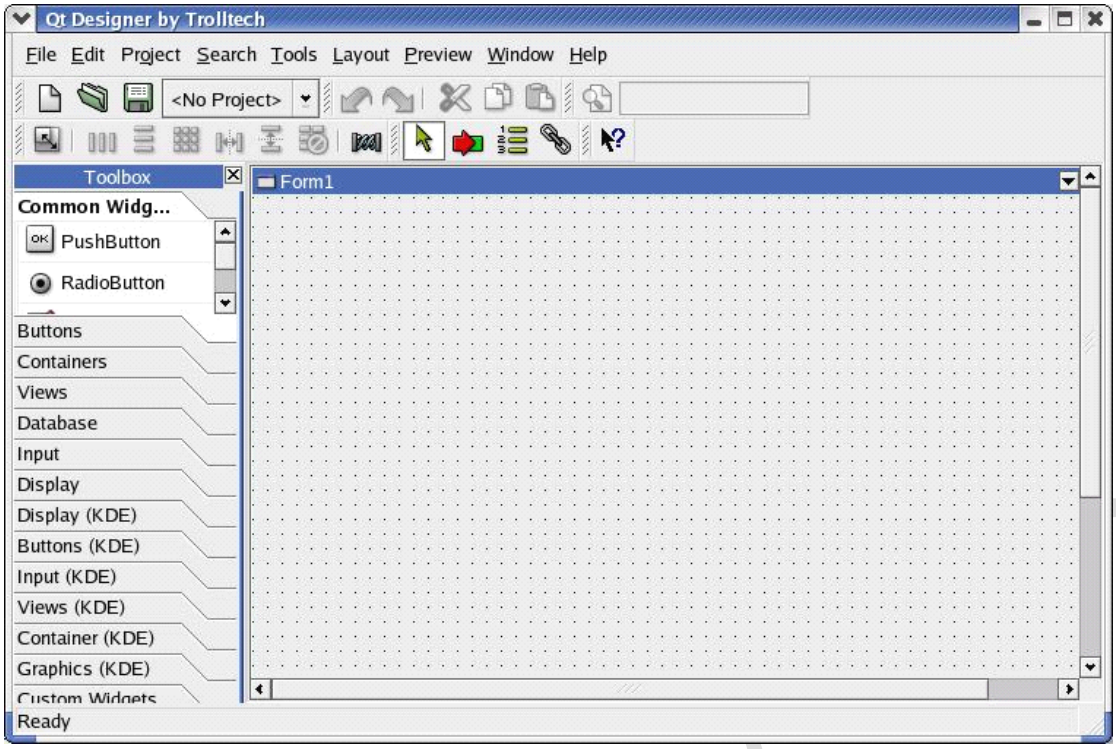


图 4-3

主窗口的左边是属性编辑器，可以在这里编辑对话框和控件的各种属性，例如标题、颜色、背景色等，不同的表单有不同的属性域。工具条下面有一些 Qt 的标准控件，可以把它们放入对话框中。先来看看对话框的两个属性。

属性 name 是表示对话框的内部名字，它不是对话框标题。回忆一下，许多 Qt 控件都把 name 作为构造函数的参数之一。可以在属性对话框中直接输入，因为那么是一个字符串变量。

另外一个属性叫 enable，表示该表单的状态，可以选择 true 或者 false，其结果是在代码中生成类似下面的语句：

```
setEnable( TRUE );
```

通过前面的学习，应该对这里提到的“属性”一词并不陌生，它是设置控件行为和外观的基本方法，只不过以前都是直接编写代码。

主窗口的大部分都被表单编辑器占据，可以在这里“画”出需要的控件。从顶部的控件列表中选择一个喜欢的控件，例如一个复选框。这些控件的外观和程序使用的控件风格有关， 所以这里看到的控件可能和最终运行的时候不大一样。可以使用鼠标任意改变控件大小或者移动它的位置。

当手工编写 Qt 应用程序的时候，需要注意控件之间的父子关系。当创建一个非顶层控 件的时候，必须指定一个父控件。在 Qt Designer 中，上面的步骤是自动的。当在某一个控 件上放置了另外一个控件后，顶层控件自动将其作为父类。可以选择窗口对象查看控件间的层次关系。

保存刚才创建的文件，Qt 把.ui 作为默认的后缀名。Qt Designer 并不直接生成 C++的源 代码，原因在于，如果用户修改了文件，那么 Qt Designer 必须修改 C++源程序，而这是一份十分困难的事情。可以打开一个.ui 文件看看，这实际上是 XML 语言格式，就好像是这个 样子：

```
<!DOCTYEP UI><UI>
<class>PizzaEntry</class>
<widget>
<class>QDialog</class>
<property>
<name>name</name>
<cstring>PizzaEntry</cstring>
</property>
……
```

这个文件是不能被编译的。必须使用另外一个工具 uic，从.ui 文件生成 C++源文件，包 括.h 文件和.cpp 文件，uic 表示 UI Compiler。例如：

```
uic –i pizza.h –o pizza.cpp pizza.ui
```

指定从文件 pizza.ui 生成头文件 pizza.h 和实现文件 pizza.cpp，现在可以查看一下生成的文件，它们都是标准的 C++格式。注意相应版本的 uic 要使用对应的 QT。

3.3.2 Tmake

Tmake 是一个很好用的生成和管理 makefile 的工具，如果不是用 tmake 来管理 makefile 的话，那将会是一件痛苦的事情，虽然有 autoconf 等工具，但毕竟还是十分繁杂的，现在 tmake 将 我们 完全 从繁 琐的生成 makefile 的 过程中 解脱出 来， 只要很简单的步骤就可以生 成 makefile 了。

在/…./qws 路径里面有各种平台支持文件和 tmake 执行文件。tmake 支持的平台有：AIX, Data General, FreeBSD, HPUX, SGI Irix, Linux, NetBSD, OpenBSD, OSF1/DEC, SCO, Solaris, SunOS, Ultrix, Unixware and Win32 用户可以根据自己的应用平台，修改其中平台支持文件。

下面我们开始学习 tmake 的使用：

我假设你有一个小的 qt 程序，他由一个 C++ header 和两个 source file 组成。首先，你要创建一个 tmake 工程文件：

progen –t app.t-o hello.pro

下面我们来产生 makefile

#tmake -o Makefile hello.pro

最后我们执行 make 命令编译 hello 这个程序。

Tmake 发行版本中有以下三个模板

App.t 用来创建生成发布使用程序的 makefile

Lib.t 用来创建生成 libraries 的 makefile

Subdirs.t 用来创建目标文档在目录中的 makefile

Tmake.conf 这个 configuration 文件包含了编译选项和各种资源库。在生成的 Makefile 文件里，如果没有达到相应的要求，可以自己手动修改，比如在编写多线程程序的时候， 就必须自己手动添加编译选项：-DQT_THREAD_SUPPORT，修改链接库：-lqte-mt。

3.3 Qt 的编程

3.3.1 准备实验嵌入式图形开发环境的搭建

实验目的：

熟练掌握如何建立嵌入式图形开发环境。

实验原理及说明：

这个实验非常重要。是一个必做的实验。这是做嵌入式图形界面程序开发所必需的开发环境。

实验步骤：

下面笔者将从以下几个步骤来介绍如何安装和使用 QTE 来进行基于 OK2440 开发目标板进行嵌入式图形的开发。以下任何步骤都经过笔者在宿主 PC 机以及 OK2440 开发目标板上进行过验证，完全可行有效。请用户您有足够的信心、耐心和细心，严格按照笔 者介绍的步骤操作。

3.3.2 准备安装软件包

cross-2.95.3.tar.bz2,

e2fsprogs-1.39.tar.gz.tar.gz，

jpegsrc.v6b.tar.gz，

libpng-1.2.18.tar.bz2，

zlib-1.2.3.tar.bz2,

qtopia-free-src-2.2.0.tar.gz

3.3.3 建立 QTE 开发环境所需要的目录

/---usr ---/local/arm

/---root---/yizhi

. . . /armlib

. . . /arminclude

. . . /qtopia-free-2.2.0

. . . /libpng-1.2.18

. . . /zlib-1.2.3

. . . /e2fsprogs-1.39

. . . /jpeg-6b

. . . /qt-2.3.10

. . . /host/qtopia-free-2.2.0

3.3.4 各个目录功能介绍

arm:存放 PC 机上交叉编译的工具

yizhi: 下载的软件包全部存放到这里

armlib: 编译软件得到 qtopia 所需要的库文件全部放在这里

arminclude: 编译软件得到 qtopia 所需要的头文件全部放在这里

qtopia-free-2.2.0: qtopia 的源代码

jpeg-6b：一个 jpeg 图形编码解码程序库

libpng-1.2.18: 一个 png 图形编码解码程序库

zlib-1.2.3: 一个压缩解压程序库

e2fsprogs-1.39: 可以提供 uuid 支持

qt-2.3.10: qt-embeddedd 的文件夹路径

host/qtopia-free-2.2.0: qtopia 的源代码

3.3.5 编译 QTOPIA 所依赖的库文件

1、 安装交叉编译工具

将 cross-2.95.3.tar.bz2 放到/usr/local/arm 文件夹下解压，

tar –zxvf cross-2.95.3.tar.bz2

这个命令会在你的/usr/local/arm/2.95.3 目录下安装 arm-linux-gcc 交叉编译程序，然后在 PATH 变量中添加一项：

export PATH=/usr/local/arm/2.95.3/bin:$PATH

测试：

arm-linux-gcc -v

2、 编译 qtopia-2.2.0 得到本机运行版本 uic 工具

qtopia-2.2.0 版本里面集成了 qt\tmake\qtopia。

编译 Qt/X11 的唯一目的就是为编译 QTE 提供 uic（用户接口编译器）以及基于 X Windows 系统的 FrameBuffer 设备模拟器 qvfb。

解压 qtopia-free-src-2.2.0.tar.gz,将 qtopia-free-2.2.0 复制到 /root/yizhi/host

cd /root/host/qtopia-free-2.2.0

要生成 Makefile

./configure

make

解压 qtopia-free-src-2.2.0.tar.gz 将其复制至 /root/yizhi/

#cp /root/yizhi/host/qtopia-free-2.2.0/qt2/bin/uic

/root/yizhi/qtopia-free-2.2.0/qt2/bin/

3、 编译 e2fsprogs-1.39

要得到 qtopia 所需的二個文件，uuid.h 及 libuuid.so。

cd /root/yizhi/e2fsprogs-1.39

设置环境变量

export PATH=/usr/local/arm/2.95.3/bin:$PATH

export CC= /usr/local/arm/2.95.3/bin/arm-linux-gcc

./configure --enable-elf-shlibs --host=arm-linux

--with-cc=/usr/local/arm/2.95.3/bin/arm-linux-gcc

--with-linker=/usr/local/arm/2.95.3/bin/arm-linux-ld

--prefix=/usr/local/arm/2.95.3/arm-linux

make

将 e2fsprogs-1.39/lib/目录下 uuid 的文件夹复制到 yizhi/arminclude 下

cp -r /lib/uuid /root/yizhi/arminclude

cp lib/libuuid.so* /root/yizhi/armlib /

4、 交叉编译 jpeg 库

解压 jpegsrc.v6b.tar.gz 到/root/yizhi

cd jpeg-6b

设置环境变量

export QTDIR=/root/yizhi/qtopia-free-2.2.0/

export QTEDIR=/root/yizhi/qtopia-free-2.2.0/qtopia/

export LD_LIBRARY_PATH=$QTDIR/lib:$LD_LIBRARY_PATH

export PATH=/usr/local/arm/2.95.3/bin:$PATH

export CC= /usr/local/arm/2.95.3/bin/arm-linux-gcc

./configure --enable-shared

修改生成的 Makefile 文件:

prefix = /root/yizhi/qt-2.3.10

CC= /usr/local/arm/2.95.3/bin/arm-linux-gcc

AR= /usr/local/arm/2.95.3/bin/arm-linux-ar rc

AR2= /usr/local/arm/2.95.3/bin/arm-linux-ranlib

保存

cp jconfig.doc jconfig.h

在你的 qt-embedded 中建立 man/man1 文件夹,否则安装 jpeg 库时会出错

mkdir -p qt-2.3.10/man/man1

最后:

make

make install –lib

报错,但是在/root/2410s/qt-2.3.10-arm/lib 目录下生成了文件夹 fonts 和 libjpeg.so*

cp jpeglib.h /root/yizhi/arminclude

cp lib/libjpeg.so* /yizhi/yizhi/armlib

就可以给 qt-embedded 添加好 for arm 的 libjpeg 库放到交叉编译的 Lib 一份

cp lib/libuuid.so* /usr/local/arm/2.95.3/arm-linux/lib

5、 编译 libpng 库:

将 libpng-1.2.18.tar.bz2 解压到/root/yizhi 目录下

cd /root/yizhi/libpng-1.2.18

cp scripts/makefile.linux ./Makefile

注意，这里的 makefile 不是用./configure 生成，而是直接从 scripts/里拷一个

vi Makefile

修改:

[begin]

...

AR_RC=/usr/local/arm/2.95.3/bin/arm-linux-ar rc

CC=/usr/local/arm/2.95.3/bin/arm-linux-gcc

RANLIB=/usr/local/arm/2.95.3/bin/arm-linux-ranlib

prefix=/usr/local/arm/2.95.3/arm-linux

...

[end]

保存

make

cp libpng12.so* /root/yizhi/armlib

cp *.h /root/yizhi/arminclude

6、 编译 zlib:

cd /root/yizhi/zlib-1.2.3

设置环境变量

export PATH=/usr/local/arm/2.95.3/bin:$PATH

export LD_LIBRARY_PATH= /usr/local/arm/2.95.3/arm-linux/lib:$LD_LIBRARY_PATH

export CC= /usr/local/arm/2.95.3/bin/arm-linux-gcc

./configure –shared

vi Makefile

修改:

[begin]

CC=/usr/local/arm/2.95.3/bin/arm-linux-gcc

...

LDSHARED=/usr/local/arm/2.95.3/bin/arm-linux-gcc -shared-Wl,-soname,libz.so.1

CPP=/usr/local/arm/2.95.3/bin/arm-linux-gcc -E

...

AR=/usr/local/arm/2.95.3/bin/arm-linux-ar rc

RANLIB=/usr/local/arm/2.95.3/bin/arm-linux-ranlib

...

prefix =/usr/local/arm/2.95.3/arm-linux

...

[end]

保存

make

将编译生成的动态库及相关的头文件分别复制到 armlib 和 arminclude 目录下

cp libz.so* /root/yizhi/armlib

cp *.h /root/yizhi/arminclude

3.3.6 交叉编译 qtopia

1、 编译修改工作

①、vi /root/yizhi/qtopia-free-2.2.0/qtopia/mkspecs/qws

/linux-arm-g++/qmake.conf

将此行

QMAKE_LIBS_QT = -lqte

修改为

QMAKE_LIBS_QT = -lqte -lpng -lz -luuid -ljpeg

②、 如果想让 Qtopia 支持鼠标，需要修改$QPEDIR/src/qt/qconfig-qpe.h 文件

注释如下部分：

/*

#ifndef QT_NO_QWS_CURSOR

#define QT_NO_QWS_CURSOR

#endif

#ifndef QT_NO_QWS_MOUSE_AUTO

#define QT_NO_QWS_MOUSE_AUTO

#endif

#ifndef QT_NO_QWS_MOUSE_PC

#define QT_NO_QWS_MOUSE_PC

#endif

*/

③、

cp /root/yuzhi/qtopia-free-2.2.0/qtopia/src/qt/Qconfig-qpe.h

/root/yizhi/qtopia-free-2.2.0/qt2/src/tools

cd $QPEDIR/src/libraries/qtopia

cp custom-linux-ipaq-g++.cpp custom-linux-arm-g++.cpp

cp custom-linux-ipaq-g++.h custom-linux-arm-g++.h

④、将/usr/local/arm/2.95.3/lib/gcc-lib/arm-linux/2.95.3/../../

../../arm-linux/sys-include/linux/videodev.h 文件中的#include<linux/

version.h>一行屏蔽

2、 设置环境变量，编译及安装

①设置编译 qtopia 的环境变量

export QTDIR=/root/yizhi/qtopia-free-2.2.0/qt2

export QPEDIR=/root/yizhi/qtopia-free-2.2.0/qtopia

export LD_LIBRARY_PATH=$QTDIR/lib:$QPEDIR/lib:$LD_LIBRARY_PATH

export TMAKEDIR=/root/yizhi/qtopia-free-2.2.0/tmake

export TMAKEPATH=$TMAKEDIR/lib/qws/linux-arm-g++

export PATH=/usr/local/arm/2.95.3/bin:$PATH

②创建交叉编译后的 qtopia 文件夹

mkdir /root/yizhi/qtopia

③生成 Makefile 文件（注释 2）

./configure -qte "-embedded -xplatform linux-arm-g++ -qconfig qpe -no-qvfb -depths 8,16,24,32 -no-xft -system-jpeg -system-libpng -system-zlib -gif -thread –release - I/root/yizhi/arminclude -L/root/yizhi/armlib -lpng -lz –luuid -ljpeg" -qpe '-xplatform linux-arm-g++ -edition pda –displaysize 640x480 -I/root/yizhi/arminclude -L/root/yizhi/armlib -prefix=/root/yizhi/qtopia'

④编译 qtopia

make

⑤安装 qtopia 的交叉编译环境

make install

编译完成以后/root/yizhi 后出现一个 qtopia 的目录，在/root/yizhi/qtopia/lib/目录下面看到 libqte.so、libqte.so.2 libqte.so.2.3、libqte.so.2.3.12 这四个文件. 这是编译 qte 的库文件.

将所有需要支持的库文件都放到新生成的 qtopia/lib 中

cp /root/yizhi/armlib/*.so* /root/yizhi/qtopia/lib

现在，您已经将 QTE 环境编译成功了，请您继续！

实验总结：

通过本实验的操作，我们已经将 QT/E 的嵌入式图形开发环境搭建完成，现在您就可以利用 QT/E 开发环境中完备的开发工具进行具体的应用程序的开发了。需要注意的是，您要时刻关注您在交叉编译应用程序时，当前终端窗口中，环境变量是否设置正确。

3.3.7 将 qtopia 挂在到目标板上

1、建立宿主机和开发板的通讯

以/root/share 为共享文件夹，目标板可通过 nfs 来访问这个共享文件夹。

将 qtopia 复制到共享文件夹/root/share/

cp -r /root/yizhi/qtopia /root/share/

设置共享文件夹：

①、修改/etc/export 指定/root/share 文件夹为共享可访问的

最后一行加上 /root/share 192.168.0.* (rw)

②、没有关闭防火墙

启动 nfs 服务

service nfs restart （或 nfs /etc/rc.d/init.d/nfs start）

设置本机的 ip 地址为 192.168.0.105

ifconfig eth0 192.168.0.105 up

2、将板子与 PC 机连接好，启动目标板，测试 qtopia 是否正常运行

配置串口的波特率为 115200，启动目标板。

minicom

ifconfig -t nfs eth0 192.168.0.16 up

检测目标板和 PC 机是否建立了连接

#ping 192.168.0.105

如果建立了连接，将文件复制到目标板上

mount 192.168.0.105:/root/share /mnt/ext2

cd /mnt/ext2/qtopia

设置环境变量

export QTDIR=$PWD

export QPEDIR=$PWD

export LD_LIBRARY_PATH=$PWD/lib

我的根文件系统为只读文件系统 cramfs，下句是指定一个可读写的文件，作为环境变量 HOME

export HOME=/etc/tmp

帧缓冲设备板子上为/dev/fb/0,但程序默认为/dev/fb0,所以做了一个链，接/dev/fb0 到/dev/fb/0

ln -s /dev/fb/0 /dev/fb0

指定键盘和鼠标为输入设备

export QWS_KEYBOARD_PROTO="USB:/dev/input/event1"

export QWS_MOUSE_PROTO="USB:/dev/input/mouse0"

在 qtopia 下可能没有 pointercal 文件可以自创一个其内容是 1 0 1 0 1 1　65536

cp pointercal /etc

cp pointercal /etc/tmp

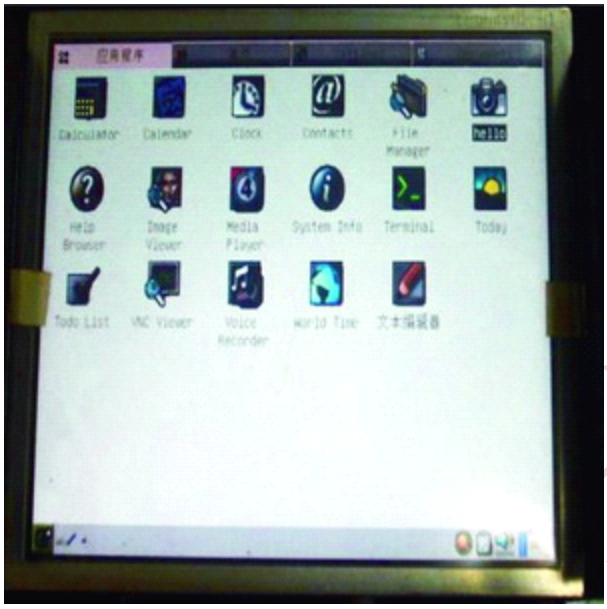
运行 PDA

cd bin

./qpe

出现 Tap anywhere on the screen to continue

用鼠标点击一下桌面，就进入配置显示使用的语言和所在地区的时区，点击 finish 键，这样就可以在显示终端上看到 qtopia 桌面环境了。



qtopia 的桌面

3.3.8 简单的“hello world”实验

实验目的：

学会如何创建和显示一个简单的窗口，了解 Qt 程序最基本的框架。

实验原理及说明：

程序见程序清单，#include <qapplication.h>包含了文件 qapplication.h，此文件总是被包含在同样的源文件中，它里面包含了 main()函数。这个例程因为使用了 QLabel widget 来显示文本，所以也必须包含文件 qlabel.h。

QApplication app(argc, argv) 创建了一个 QApplication 对象，命名为 app。QApplication 对象是一个容器，包含了应用程序顶层的窗口（或者一组窗口）。顶层窗口是独一无二的，它在应用程序中从来就没有父窗口。因为 QApplication 对象的任务是控制管理你的应用程 序，因此在每个应用程序中只能有一个 QApplication 对象。此外，由于创建对象的过程必须 初始化 Qt 系统，所以在使用其它任何 Qt 工具之前，QApplication 对象必然已经存在了。

一个 Qt 程序就是一个标准的 C++程序。这就意味着为了启动程序，函数 main()将被操作系统所调用。而且，像所有的 C++程序一样，命令行选项可能会也可能不会传递给 main() 函数。命令行选项作为初始化过程的一部分传递给 Qt 系统，也是体现在 QApplication app(argc, argv)这条语句中。

两个命令行参数：argc 和 argv，在 app 结构中使用，因为这样可以指定一些特殊的标志和设置。例如，用-geometry 参数启动 Qt 程序，将可以指定窗口显示的位置和大小。通过修改启动程序的外观信息，用户可以自己按照喜好定义程序的外观风格。

QLabel *label = new QLabel("Hello, world!", 0);创建了一个 QLabel 部件。QLabel 部 件是一个简单的窗口，能够显示一个字符串。标签指定的父类部件创建为 0,因为这个标签将 被作为顶层的窗口，而顶层窗口是没有父类的。Qlabeld 类有三个被定义的构造函数，如下 所示，其具体定义参见相关文档

QLabel (QWidget * parent, const char * name = 0, WFlags f = 0)

QLabel(const QString&text, QWidget*parent, const char*name = 0, WFlags f = 0) QLabel(QWidget*buddy, constQString&text, QWidget*parent, const char*name = 0,WFlags f = 0)

其中 text 参数即为 QLabel 对象所要显示的文本信息，在本程序构造 QLabel 对象时，传入的参数为“Hello， world!”。

QLabel 默认的动作是以垂直方向中心对齐的方式显示字符串，以左边为基准。label->setAlignment(Qt::AlignVCenter | Qt::AlignHCenter) 调用 setAlignment()函数使得 文本在水平和垂直方向上都是位于中心位置。

label->setGeometry(10, 10, 200, 80)决定了标签部件在 QApplication 窗口中的位置、高度和宽度。其中 setGeometry()函数的原形为：setGeometry (int x, int y, int w, int h)。应此 label 在 QApplication 窗口中的坐标是（10， 10）。宽度为 200，高度为 80。

app.setMainWidget(label)语句的作用是把 QLabel 所定义的对象插入到主窗口中。通 常插入的主窗口的的部件应该是某种复合部件，是多个部件、文本以及其它应用程序主窗口 元件的集合。在本例程中，为了简单起见，插入的对象只是一个简单的标签部件。

label->show()所起的作用是在实现标签在窗口上的显示所必需的。Show()函数并不立刻显示 widget,它只是显示做好准备，以便需要的时候能够显示出来。在这个例程中，父窗口，即 QApplication 窗口，负责显示标签，但它只是在调用标签的 show()方法时才会完成显示。

另外一个函数（），叫做 hide()，用于使一个部件从屏幕上消失。

int result = app.exec()；return result 两条语句调用了 exec()函数和返回系统一个值。 exec()函数只要当程序停止执行的才返回，它返回一个整型的值，代表程序的完成状态，由 于我们并不处理状态代码，这个值只是被简单的返回给系统。

这个例子比较简单，只有一个源文件组成，因此编译它的 Makefile 文件比较简单。但 我们不打算自己编写 Makefile，而是采用 tmake 工具生成 Makefile 以减少编写程序的困难。

在 Makefile 文件中，Makefile 认为环境变量 QTDIR 已经被定义好了，它被用来指明 Qt 开发系统的安装路径。通常，这个环境变量在安装软件的时候就已经配置好它的定义了。因 此，在用户编译之前，请务必检查一下环境变量是否正确。

在 Makefile 文件中，定义了一些常用的变量。这些变量有必要解释一下以利于以后的 Makefile 的阅读。如下表所示：

Makefile 中定义的常见变量	
名称	内容
INCL	定义头文件的路径。它们被传递给编译器，告诉编译器到何处去寻找
CFLAGS	定义传给编译器的选择项列表。例如：-pipe 选项指示编译器在两个编译平台间传输数据时使用管道，而不是临时文件。-O2 选项设置了一个相当高的最优化级别。
LFLAGS	定义传给连接器的选择项列表。每个-L 选项所设置得是包含一个或者
LIBS	定义这个程序需要用到的库的名称列表，将会在 LFLAGS 指明的路径下寻找这些库。这些名称都被扩展称库的名字。例如，-lqte 将改变为 libqte.so，-lm 则成为 libm.so
CC	定义 C 编译器的名字
C××	定义 C++编译器的名字

实验步骤：

1、 硬件连接：

按照前面文档说明连接宿主 PC 机和一台 OK2440 开发板。

2、在宿主 PC 机端，打开一个终端窗口，启动终端窗口，我们必须为该终端窗口重新设置一些重要的环境变量（如：QTDIR、QTEDIR、 TMAKEPATH、 LD_LIBRARY_PATH 、PATH 这 5 个环境变量），注意这个过程非常重要， 请您输入下列几条命令：

```
# export QTDIR=/root/yizhi/qtopia-free-2.2.0/qt2
```

export QPEDIR=/root/yizhi/qtopia-free-2.2.0/qtopia

export LD_LIBRARY_PATH=$QTDIR/lib:$QPEDIR/lib:$LD_LIBRARY_PATH

export TMAKEDIR=/root/yizhi/qtopia-free-2.2.0/tmake

export TMAKEPATH=$TMAKEDIR/lib/qws/linux-arm-g++

export PATH=/usr/local/arm/2.95.3/bin:$PATH

上述 export 命令输入完成后， 我们就可以开始编译 QTE 实例了。 在上面的打开的同一个终端窗口中(注：一定要在上面已经重新设置了环境变量的同一个终端窗口下来编译 QTE 实例)，请您输入下列命令编译 QTE 实例 qt1，其编译过程主 要有四大步，分别是：

◆生成工程文件（*.pro）

◆生成 Makefile 文件

◆修改生成的 Makefile 文件

◆编译生成目标文件

designer

保存生成 Form1.ui 文件，并依赖此文件生成 .h 和 .cpp 文件

uic Form1.ui –o hello.h

uic –impl hello.h Form1.ui –o hello.cpp

打开 hello.cpp 文件，用 QT 的信号与槽通讯机制，将设计的 designer 界面上的控件关联起来，再自己编写好 hello.cpp 和 hello.pro 文件

tmake hello.pro –o Makefile

编译运行

make

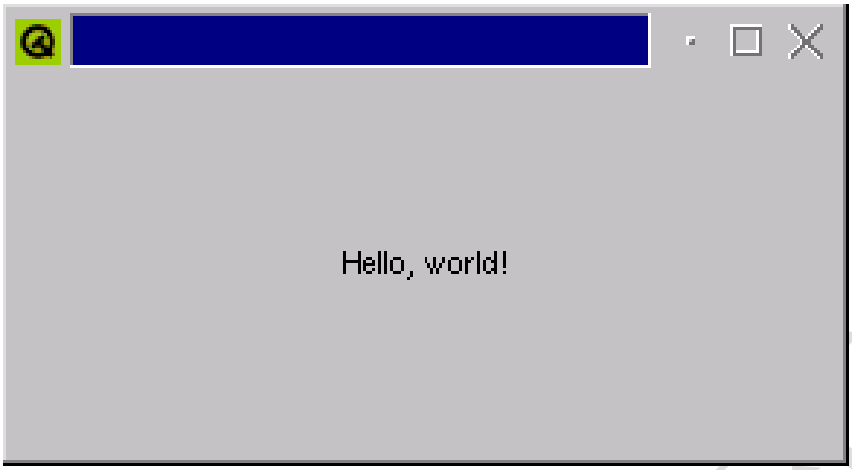
就生成了可执行的文件 hello 了

3、将可执行文件挂载到目标板上进行检测

然后将可执行的 hello 文件复制到/root/yizhi/qtopia/bin 文件下，然后重复上述 3.4 的步骤，将 qtopia 文件再挂载到目标板上，检测一下是否运行顺利。如果顺利就将 qtopia 做成文件系统，通过 DNW 烧写到目标板的 flash 里，这样重新启动目标板，就能看到自己做的图像了。

4、将做好的程序下载到目标板上

将 qtopia 做成文件系统通过 DNW 软件下载到目标板的 flash 中，当上电运行 linux 时，包括自己编写的应用程序的 qtopia 界面就可以出现了。



实验参考程序：

helloword.cpp:　　程序清单 1.1

```
*********************************************************************************
#include <qapplication.h>
#include <qlabel.h>
int main( int argc, char **argv )
{
QApplication app( argc, argv );
QLabel *label = new QLabel( "Hello, world!", 0 );
label->setAlignment( Qt::AlignVCenter | Qt::AlignHCenter );
label->setGeometry( 10, 10, 200, 80 );
app.setMainWidget( label );
label->show();
int result = app.exec();
return result;
}
*********************************************************************************
```

实验总结：

本实验主要给了一个用户最初的关于 Qt 应用程序的框架，使用户对 Qt 程序的编写有一个初步的印象。

附录一 Minicom 使用指南

minicom 是 Linux 下的一个友好的串口通信程序，类似于 Windows 操作系统上超级终端工具。

总览 SYNOPSIS

minicom [-somMlwz8] [-c on|off] [-S script] [-d entry] [-a on|off] [-t term] [-p pty] [-C capturefile] [*configuration*]

描述 DESCRIPTION

minicom 是个通信程序，有点象共享软件 TELIX，但其源码可以自由获得，并能够运行于多数 Unix 系统。它包括以下特性：自动重拨号的拨号目录，对串行设备 UUCP 格式的 lock 文件的支持，独立的脚本

语言解释器，文件捕获，多用户单独配置，等等。

命令行参数 COMMAND-LINE

-s 设置，root 使用此选项在/etc/minirc.dfl 中编辑系统范围的缺省值。使用此参数后，minicom 将不 进行初始化，而是直接进入配置菜单。如果因为你的系统被改变，或者第一次运行 minicom 时，minicom 不能启动，这个参数就会很有用。对于多数系统，已经内定了比较合适的缺省值。

-o 不进行初始化，minicom 将跳过初始化代码。如果未复位(reset)就退出了 minicom，又想重启一次会话(session)，那么用这个选项就比较好(不会再有错误提示：modem is locked --注)。但是也有潜在的危险：由于未对 lock 文件等进行检查，因此一般用户可能会与 uucp 之类的东东发生冲突.....也许以后这个参数会被去掉。现在姑且假定，使用 modem 的用户对自己的行为足够负责。

-m 用 Meta 或 Alt 键重载命令键，在 1.80 版中这是缺省值，也可以在 minicom 菜单中配置这个选项。不过若一直使用不同的终端，其中有些没有 Meta 或 Alt 键，那么方便的做法还是把缺省的命令键设置为

Ctrl－A，当有了支持 Meta 或 Alt 键的键盘时再使用此选项。Minicom 假定 Meta 键发送 ESC 前缀，而不是设置字符最高位的那一种(见下)。

－M 跟“－m”一样，但是假定 Meta 键设置字符高端的第八位(发送 128+字符代码)。

－z 使用终端状态行。仅当终端支持，并且在其 termcap 或 terminfo 数据库入口中有相关信息时才可用。

－l 逐字翻译高位被置位的字符，使用此标志，minicom 将不再尝试将 IBM 行字符翻译为 ASCII 码，而是将其直接传送。许多 PC－Unix 克隆不经翻译也能正确显示它们(Linux 使用专门的模式：Coherent 和 Sco)。

－a 特性使用，有些终端，特别是 televideo 终端，有个很讨厌的特性处理(串行而非并行)。minicom 缺省使用‘－a on’，但若在用这样的终端，你就可以(必须!)加上选项‘－a off’。尾字‘on’或‘off’需要加上。

－t 终端类型，使用此标志，可以重载环境变量 TERM，这在环境变量 MINICOM 中使用很方便；可以创建一个专门的 termcap 入口以备 minicom 在控制台上使用，它将屏幕初始化为 raw 模式，这样，连同‘－f’标志一起，就可以不经翻译而显示 IBM 行字符。

－c 颜色使用，有些终端(如 Linux 控制台)支持标准 ANSI 转义序列色彩。由于 termcap 显然没有对于色彩的支持，因而 minicom 硬性内置了这些转义序列的代码。所以此选项缺省为 off。使用‘－c on’可以打开此项。把这个标志，还有‘－m’放入 MINICOM 环境变量中是个不错的选择。

－S 脚本，启动时执行给定名字的脚本。到目前为止，还不支持将用户名和口令传送给启动脚本。如果还使用了‘－d’选项，以在启动时开始拨号，此脚本将在拨号之前运行，拨号项目入口由‘－d’指明。

－d 启动时拨打拨号目录中的一项，可以用索引号知名，也可以使用入口项的一个子串。所有其它程序初始化过程结束后，拨号将会开始。

－p 要使用的伪终端，它超载配置文件中定义的终端端口，但仅当其为伪 tty 设备。提供的文件名必须采用这样的形式：(/dev/)tty[p－z][0－f]

－C 文件名，启动时打开捕获文件。

－8 不经修改地传送 8 位字符，“连续”意指未对地点/特性进行真正改变，就不插入地点/特性控制序列。此模式用于显示 8 位多字节字符，不是 8 位自否的语言都需要（例如显示芬兰文字就不需要这个）。minicom 启动时，它首先搜索用于命令行参数的 MINICOM 环境变量，这些参数可在命令行上超载。例如：若进行了如下设置：

```
MINICOM='－m －c on'
```

```
export MINICOM
```

或者其它等效的设置，然后启动 minicom，minicom 会假定终端有 Meat 键或 Alt 键，并且支持彩色。如果从一个不支持彩色的终端登录，并在启动文件(.profile 或等效文件)中设置了 minicom，而且又不想重置环境变量，那么就可以键入‘minicom -c off’，来运行这次没有色彩支持的会话。

使用 USE

Minicom 是基于窗口的。要弹出所需功能的窗口，可按下 Ctrl-A (以下使用 C-A 来表示 Ctrl-A)，然后再按各功能键(a-z 或 A-Z)。先按 C-A，再按’z’，将出现一个帮助窗口，提供了所有命令的简述。配置 minicom(-s 选项，或者 C-A、O)时，可以改变这个转义键，不过现在还是用 Ctrl-A。

以下键在所有菜单中都可用：

```
UP arrow - up 或 ’k’
DOWN arrow - down 或 ’j’
LEFT arrow - left 或 ’h’
RIGHT arrow - right 或 ’l’
CHOOSE Enter
CANCEL ESCape.
```

屏幕分为两部分：上部 24 行为终端模拟器的屏幕。ANSI 或 VT100 转义序列在此窗口中被解释。若底部还剩有一行，那么状态行就放在这儿；否则，每次按 C-A 时状态行出现。在那些有专门状态行的终端上将会使用这一行，如果 termcap 信息完整且加了 -k 标志的话。

下面按字母顺序列出可用的命令：

C-A 两次按下 C-A 将发送一个 C-A 命令到远程系统。如果把“转义字符”换成了 C-A 以外的什么字符，则对该字符的工作方式也类似。

A 切换“Add Linefeed”为 on/off。若为 on，则每上回车键在屏幕上显示之前，都要加上一个 linefeed。

B 提供一个回卷(scroll back)的缓冲区。可以按 u 上卷，按 d 下卷，按 b 上翻一页，按 f 下翻一页。也可用箭头键和翻页键。可用 s 或 S 键(大小写敏感)在缓冲区中查找文字串，按 N 键查找该串的下一

次出现。按 c 进入引用模式，出现文字光标，就可以按 Enter 键指定起始行。然后回卷模式将会结束，带有前缀'>'的内容将被发送。

C　清屏。

D　拨一个号，或转向拨号目录。

E　切换本地回显为 on/off （若你的 minicom 版本支持）。

F　将 break 信号送 modem。

G　运行脚本(Go)。运行一个登录脚本。

H　挂断。

I　切换光标键在普通和应用模式间发送的转义序列的类型(另参下面 关于状态行的注释)。

J　跳至 shell。返回时，整个屏幕将被刷新(redrawn)。

K　清屏，运行 kermit，返回时刷新屏幕。

L　文件捕获开关。打开时，所有到屏幕的输出也将被捕获到文件中。

M　发送 modem 初始化串。若你 online，且 DCD 线设为 on，则 modem 被初始化前将要求你进行确认。

O　配置 minicom。转到配置菜单。

P　通信参数。允许你改变 bps 速率，奇偶校验和位数。

Q　不复位 modem 就退出 minicom。如果改变了 macros，而且未存盘，会提供你一个 save 的机会。

R　接收文件。从各种协议(外部)中进行选择。若 filename 选择窗口和下载目录提示可用，会出现一个要求选择下载目录的窗口。否则将使用 Filenames and Paths 菜单中定义的下载目录。

S　发送文件。选择你在接收命令中使用的协议。如果你未使文件名选择窗口可用(在 File Transfer Protocols 菜单中设置)，你将只能在一个对话框窗口中写文件名。若将其设为可用，将弹出一个窗口，显示你的上传目录中的文件名。可用空格键为文件名加上或取消标记，用光标键或 j/k 键上下移动光标。被选的文件名将高亮显示。目录名在方括号中显示，两次按下空格键可以在目录树中上下移动。最后，按 Enter 发送文件，或按 ESC 键退出。

T　选择终端模拟：ANSI(彩色)或 VT100。此处还可改变退格键，打开或关闭状态行。

W　切换 linewrap 为 on/off。

X　退出 minicom，复位 modem。如果改变了 macros，而且未存盘，会提供一个 save 的机会。

Z　弹出 help 屏幕。

拨号目录 DIALING DIRECTORY

按下 Ctrl－A、D，会进入拨号目录。可以增减、删除或修改各个项目。选择 “dial”，则会拨打标记项目的电话号码，或者当未作任何标记时高亮显示的项目号码。modem 拨号时，可按 ESC 取消；任何其它按键将关闭拨号窗口，但并不取消拨号。拨号目录将保存在 home 目录下的“.dialdir” 文件中。可用箭头键可以上下卷动，但也可用 PageUp 或 PageDown 键卷动整页。若没有这些键，可用 Ctrl－B(向后)，以及 Ctrl－F(向前)。可用空格键标记多个项目，若 minicom 不能建立一个连接，它将在此列表中循环进行拨号。目录中标记项目的名字前将显示一个‘>’符号。

“edit” 菜单不言自明，但这里还是简要介绍一下。

A － Name 项目名

B － Number 电话号码

C － Dial string # 指出用于连接的拨号串。在 Modem and dialing 菜单中有三种不同的拨号串(前缀和后缀)可以进行设置。

D － Local echo 可为 on 或 off（若你的 minicom 版本支持）

E － Script 成功建立连接后必须执行的脚本(参 runscript 手册)

F － Username 传给 runscript 程序的用户名。在环境串“$LOGIN”中传送。

G － Password 传送为“$PASS”的口令。

H － Terminal Emulation 使用 ANSI 或 VT100 模拟。

I － Backspace key sends 退格键发送的代码(Backspace 或 Delete)。

J － Linewrap 可为 on 或 off。

K － Line settings 本次连接的 bps 速率，位数和奇偶设置。速率可选当前值，这样就能用当时正在使用的任何速率值。

L － Conversion table 可以指定运行 login 脚本前，此拨号项目应答的任何时候要装入的字符转换表。若此域为空，则转换表保持不变。

edit 菜单还显示了最近一次呼叫此项的日期和时间，及呼叫该项的总次数。但并不允许改变这些值。当进行连接时，它们会自动更新。

配置 CONFIGURATION

通常，minicom 从文件“minirc.dfl”中获取其缺省值。不过，若给 minicom 一个参数，它将尝试从文件“minirc.configuration”中获取缺省值。因此，为不同端口、不同用户等创建多个配置文件是可能的。最好使用设备名，如：tty1，tty64，sio2 等。如果用户创建了自己的配置文件，那么该文件将以“.minirc.dfl”为名出现在 home 目录中。

按 Ctrl－A、O，进入 setup 菜单。人人都可以改变其中的多数设置，但有些仅限于 root。在此，那些特权设置用星号(*)标记。

Filenames and paths 此菜单定义缺省目录。

A - Download directory 下载的文件的存放位置

B - Upload directory 从此处读取上传的文件

C - Script directory 存放 login 脚本的位置

D - Script program 作为脚本解释器的程序。缺省是“runscript”，也可用其它的东西(如: /bin/sh 或 “expect”)。Stdin 和 Stdout 连接到 modem，Stderr 连接到屏幕。若用相对路径（即不以’/’开头），则是相对于 home 目录，除了脚本解释器以外。

E - Kermit program 为 kermit 寻找可执行程序和参数的位置。命令行上可用一些简单的宏：‘%l’扩展为拨出设备的完整文件名，‘%b’扩展为当前波特率。

F - Logging options Options to configure the logfile writing.

File Transfer Protocols 此处规定的协议将在按下 Ctrl－A、s/r 时显示。行首的 “Name”为将要显示在菜单中的名字。“Program”为协议路径，其后的“Name”则确定了程序是否需要参数，如要传送的文件。“U/D”确定了该项要否在“upload/download”菜单中出现。“Fullscr”确定要否全屏运行，否则 mincom 将仅在一个窗口中显示其标准输出。“IO－Red” 确定 minicom 要否将程序的标准 io 连接到 modem 端口。“Multi”告诉文件名选择窗口协议能否用一个命令发送多上文件。它对于下载协议无效；如果不用文件名选择窗口，那么上传协议也会忽略它。老版本的 sz 和 rz 非全屏，并且设置了 IO－Red。但是，有些基于 curses 的版本，至少是 rz，不希望其 stdin 和 stdout 被改向，以及全屏运行。所有文件传输协议都以用户的 UID 运行，但并不是总有 UID=root。对于 kermit，命令行上可用‘%l’和‘%b’。在此菜单内，还能规定当提示文件要上传时，要否文件选择窗口，以及每次自动下载开始时要否提示下载目录。如果禁止下载目录提示，将使用 file and directory 菜单中规定的下载目录。

附录二 Linux 常用命令详解

Linux 是一个真正的多用户操作系统，它可以同时接受多个用户登录。Linux 还允许一个用户进行多次登录，这是因为 Linux 和 UNIX 一样，提供了虚拟控制台的访问方式，允许用户在同一时间从控制台进行多次登录。虚拟控制台的选择可以通过按下 Alt 键和一个功能键来实现，通常使用 F1－F6 例如，用户登录后，按一下 Alt－F2 键，用户又可以看到"login:"提示符，说明用户看到了第二个虚拟控制台。然后只需按 Alt－F1 键，就可以回到第一个虚拟控制台。一个新安装的 Linux 系统默认允许用户使用 Alt－F1 到 Alt－F6 键来访问前六个虚拟控制台。虚拟控制台可使用户同时在多个控制台上工作，真正体现 Linux 系统多用户的特性。用户可以在某一虚拟控制台上进行的工作尚未结束时，切换到另一虚拟控制台开始另一项工作。

当然我们也可以在 KDE 环境下使用终端方式输入命令。常见的命令如下：

文件列表 － ls

ls #以默认方式显示当前目录文件列表

ls －a #显示所有文件包括隐藏文件

ls －l #显示文件属性，包括大小，日期，符号连接，是否可读写及是否可执行

ls －-color=never *.so > obj #不显示文字颜色,将所有 so 文件记录到 obj 文件中

目录切换 － cd

cd dir #切换到当前目录下的 dir 目录

cd / #切换到根目录

cd .. #切换到到上一级目录

cd ../.. #切换到上二级目录

cd ~ #切换到用户目录，比如是 root 用户，则切换到/root 下

删除 － rm

rm file #删除某一个文件

rm －fr dir #删除当前目录下叫 dir 的整个目录

复制 － cp

cp source target #将文件 source 复制为 target

cp /root/source . #将/root 下的文件 source 复制到当前目录

cp -av soure_dir target_dir #将整个目录复制，两目录完全一样

cp -fr source_dir target_dir #将整个目录复制，并且是以非链接方式复制，当 source 目录带有符号链接时，两个目录不相同

移动 - mv

mv source target #将文件 source 更名为 target

比较 - diff

diff dir1 dir2 #比较目录 1 与目录 2 的文件列表是否相同，但不比较文件的实际内容，不同则列出

diff file1 file2 #比较文件 1 与文件 2 的内容是否相同，如果是文本格式的文件，则将不相同的内容显示，如果是二进制代码则只表示两个文件是不同的

comm file1 file2 #比较文件，显示两个文件不相同的内容

回显 - echo

echo message #显示一串字符

echo “essage message2”#显示不连续的字符串

文件内容查看 - cat

cat file #显示文件的内容，和 DOS 的 type 相同

cat file | more #显示文件的内容并传输到 more 程序实现分页显示，使用命令 less file 可实现相同的功能

more #分页命令，一般通过管道将内容传给它，如 ls | more

设置环境变量 - export

export LC_ALL=zh_CN.GB2312 #将环境变量 LC_ALL 的值设为 zh_CN.GB2312

export DISPLAY=0:0 #通过该设置，当前字符终端下运行的图形程序可直接运行于 Xserver

时间日期 - date

date #显示当前日期时间

date -s 20:30:30 #设置系统时间为 20:30:30

date -s 2002-3-5 #设置系统时期为 2003-3-5

clock -r #对系统 Bios 中读取时间参数

clock -w #将系统时间(如由 date 设置的时间)写入 Bios

容量查看 - du

du #计算当前目录的容量

du -sm /root #计算/root 目录的容量并以 M 为单位

查找 - find

find -name /path file #在/path 目录下查找看是否有文件 file

搜索 - grep

grep -ir “chars” #在当前目录的所有文件查找字串 chars，并忽略大小写，-i 为大小写，-r 为下一级目录

编辑 - vi

vi file #编辑文件 file

vi 原基本使用及命令：

输入命令的方式为先按 ctrl+c，然后输入:x(退出),:x!(退出并保存):w(写入文件),:w!(不询问方式写入文件)，:r file(读文件 file)，:%s/oldchars/newchars/g(将所有字串 oldchars 换成 newchars)这一类的命令进行操作

读取 - man

man ls #读取关于 ls 命令的帮助

man ls | grep color #读取关于 ls 命令的帮助并通过 grep 程序在其中查找 color 字串

重启 - reboot

reboot #重新启动计算机

关机 - halt

halt #关闭计算机

压缩与解压 - tar

tar xfzv file.tgz #将文件 file.tgz 解压

tar xfzv file.tgz -C target_path #将文件 file.tgz 解压到 target_path 目录下

tar cfzv file.tgz source_path #将文件 source_path 压缩为 file.tgz

tar c directory > directory.tar #将目录 directory 打包成不压缩的 directory.tar

gzip directory.tar #将覆盖原文件生成压缩的 directory.tar.gz

gunzip directory.tar.gz #覆盖原文件解压生成不压缩的 directory.tar。

tar xf directory.tar #可将不压缩的文件解包

权限设置 -chmod

chmod a+x file #将 file 文件设置为可执行，脚本类文件一定要这样设置一个，否则得用 bash file 才能执行

chmod 666 file #将文件 file 设置为可读写

chown user /dir #将/dir 目录设置为 user 所有

网卡配置 - ifconfig

ifconfig eth0 192.168.1.1 netmask 255.255.255.0 #设置网卡 1 的地址 192.168.1.1，掩码为 255.255.255.0，不写 netmask 参数则默认为 255.255.255.0

ifconfig eth0:1 192.168.1.2 #捆绑网卡 1 的第二个地址为 192.168.1.2

ifconfig eth0:x 192.168.1.x #捆绑网卡 1 的第二个地址为 192.168.1.x

ifconfig down eth1 #关闭第二块网卡，使其停止工作

创建设备 - mknod

mknod /dev/hda1 b 3 1 #创建块设备 hda1，主设备号为 3，从设备号为 1，即 master 硬盘的的第一个分区

mknod /dev/tty1 c 4 1 #创建字符设备 tty1,主设备号为 4，众设备号为 1，即第一个 tty 终端

装载模块 - insmod

insmod rtl8139.o #装载驱动程序 rtl8139.o

insmod sb.o io=0x280 irq=7 dma=3 dma16=7 mpu_io=330 #装载驱动程序并设置相关的 irq,dma 参数

删除模块 - rmmod

rmmod rtl8139 #删除名为 rtl8139 的驱动模块

挂接 - mount

mount -t ext2 /dev/hda1 /mnt #把/dev/hda1 装载到 /mnt 目录

mount -t iso9660 /dev/cdrom /mnt/cdrom #将光驱加载到/mnt/cdrom 目录

mount -t smb //192.168.1.5/sharedir /mnt -o username=fangtan,password =fangtan #将 windows 的的共享目录加载到/mnt/smb 目录，用户名及密码均为 fangtan

mount -t nfs 192.168.1.1:/sharedir /mnt #将 nfs 服务的共享目录 sharedir 加载到/mnt/nfs 目录

卸载 - umount

umount /mnt #将/mnt 目录卸载，/mnt 目录必须处于空闲状态

umount /dev/hda1 #将/dev/hda1 设备卸载，设备必须处于空亲状态

进程查看 - ps

ps #显示当前系统进程信息

ps -ef #显示系统所有进程信息

杀死进程 - kill

kill -9 500 #将进程编号为 500 的程序杀死

飞凌嵌入式

附录三 内核配置详细说明

第一部分

Code maturity level options ---> 代码成熟等级选项

[ ]Prompt for development and/or incomplete code/drivers

默认情况下是选择的，这将会在设置界面中显示还在开发或者还没有完成的代码与驱动.你应该选择它，因为有许多设备可能必需选择这个选项才能进行配置，实际上它是安全的。

[ ] Select only drivers expected to compile cleanly

选择这个选项你将不会看到一些已知的存在问题的驱动程序选项，默认的情况下也是选择的。如果你有设备没有找到驱动选项，你可以将这一项去掉，或许就可以找到相关驱动了，不过它可能是有 BUG 的。

第二部分

General setup ---> 常规安装选项

Local version - append to kernel release

[*] Support for paging of anonymous memory (swap)

这个选项将使你的内核支持虚拟内存，也就是让你的计算机好象拥有比实际内存更多 多的内存空间用来执行很大的程序。默认是选择的。

[*] System V IPC

是组系统调用及函数库，它能让程序彼此间同步进行交换信息。某些程序以及 DOS 模拟环境都需要它。为进程提供通信机制，这将使系统中各进程间有交换信息与保持同步的能力。有些程序只有在选 Y 的情况下才能运行，所以不用考虑，这里一定要选。

[ ] POSIX Message Queues

POSIX 消息队列,这是 POSIX IPC 中的一部分

[ ] BSD Process Accounting

一般用户所执行的程序，可通知内核将程序统计数据写成文件，详细记录相关信息。

[*] Sysctl support

此项功能可在不重新编译内核或是重新开机的前提下，动态变更某些特定的内核参数和变量，赋予内核较

大的弹性。除非内存太少或是编译出来的内核是给安装、救援磁盘所使用，否则这个选项一定要选上。这将提供一个接口让你可以动态的更改一些核心参数与变量，而不需要重新启动系统。打开这个选项将会增加内核的体积至少 8KB。如果你的内核仅用制作安装与恢复系统系统盘那么可以不选，以减少对内存的占用。

[ ] Auditing support

审记支持，用于和内核的某些子模块同时工作，例如 SELinux。只有选择此项及它的子项，才能调用有关审记的系统调用。

--- Support for hot-pluggable devices

是否支持热插拔的选项，肯定要选上。不然 USB、PCMCIA 等这些设备都用不了。

[*] Kernel Userspace Events

内核中分为系统区和用户区，这里系统区和用户区进行通讯的一种方式，选上。

[ ] Kernel .config support

这将会把内核的配置信息与相关的文档说明编译进内核中，以后可以使用一些工具来提取它用来重新构建内核，一般不用选它。

[ ] Configure standard kernel features (for small systems) --->

这是为了编译某些特殊用途的内核使用的，例如引导盘系统。通常你可以不选择这一选项，你也不用对它下面的子项操心了。

第三部分

Loadable module support ---> 可引导模块支持 建议作为模块加入内核

[*] Enable loadable module support

这个选项可以让你的内核支持模块，模块是什么呢？模块是一小段代码，编译后可在系统内核运行时动态的加入内核，从而为内核增加一些特性或是对某种硬件进行支持。一般一些不常用到的驱动或特性可以编译为模块以减少内核的体积。在运行时可以使用 modprobe 命令来加载它到内核中去(在不需要时还可以移除它)。一些特性是否编译为模块的原则是，不常使用的，特别是在系统启动时不需要的驱动可以将其编译为模块，如果是一些在系统启动时就要用到的驱动比如说文件系统，系统总线的支持就不要编为模块了，否在无法启动系统。

[*] Module unloading

这个选项可以让你卸载不再使用的模块，如果不选的话你将不能卸载任何模块(有些模块一旦加载就不能卸载，不管是否选择了这个选项)。不选择这个选项会让你的内核体积减小一点。

[*] Forced module unloading

允许强制卸载正在使用中的模块(比较危险)

[ ] Module versioning support (EXPERIMENTAL)

允许使用其他内核版本的模块(可能会出问题)

[ ] Source checksum for all modules

这个功能是为了防止更改了内核模块的代码但忘记更改版本号而造成版本冲突。我估计现在没有哪家公司在开发中还没使用版本控制工具，所以不需要这项了。如果你不是自己写内核模块，那就更不需要这一选项了。

[*] Automatic kernel module loading

一般情况下，如果我们的内核在某些任务中要使用一些被编译为模块的驱动或特性时，我们要先使用 modprobe 命令来加载它，内核才能使用。不过，如果你选择了这个选项，在内核需要一些模块时它可以自动调用 modprobe 命令来加载需要的模块，这是个很棒的特性，当然要选 Y 喽。

第四部分

System Type ---> 系统类型

ARM system type (Samsung S3C2410) ---> ARM 系统，基于 S3C2410

S3C24XX Implementations ---> 基于 S3C24xx 架构的实现

--- S3C2410 Boot

--- S3C2410 Setup

[*] S3C2410 DMA support 支持 DM 功能

[ ] S3C2410 DMA support debug 支持 DMA 调试功能

(0) S3C2410 UART to use for low-level messages

--- Processor Type 处理器类型

--- Processor Features 处理器特性

[*] Support Thumb user binaries 支持 Thumb 指令集

[ ] Disable I-Cache 禁止指令缓存（Instruction Cache，I-Cache）。

[ ] Disable D-Cache 禁止数据缓存（Data Cache，D-Cache）

[ ] Force write through D-cache 强制回写数据缓存

第五部分

Bus support --->总线支持

PCCARD (PCMCIA/CardBus) support --->

< > PCCard (PCMCIA/CardBus) support

PCMCIA 卡支持

第六部分

Kernel Features

[ ] Symmetric Multi-Processing (EXPERIMENTAL)

[ ] Preemptible Kernel (EXPERIMENTAL)

第七部分

Boot options --->

(0x0) Compressed ROM boot loader base address

(0x0) Compressed ROM boot loader BSS address

(console=ttyS0,115200) Default kernel command string

[ ] Kernel Execute-In-Place from ROM

第八部分

Floating point emulation --->

--- At least one emulation must be selected

[*] NWFPE math emulation

[ ] Support extended precision

[ ] FastFPE math emulation (EXPERIMENTAL)

第九部分

Userspace binary formats ---> 支持的可执行文件格式

[*] Kernel support for ELF binaries

ELF 是开放平台下最常用的二进制文件格式,支持动态连接,支持不同的硬件平台.除非你知道自己在做什么,否则必选

< > Kernel support for a.out and ECOFF binaries

这是早期 UNIX 系统的可执行文件格式，目前已经被 ELF 格式取代。

< > Kernel support for MISC binaries

允许插入二进制的封装层到内核中,使用 Java,.NET,Python,Lisp 等语言编写的程序时需要它

< > RISC OS personality

第十部分

Power management options ---> 电源管理选项

[ ] Power Management support

电源管理有 APM 和 ACPI 两种标准且不能同时使用.即使关闭该选项,X86 上运行的 Linux 也会在空闲时发出 HLT 指令将 CPU 进入睡眠状态

第十一部分

Device Drivers --->设备驱动程序

Generic Driver Options --->

驱动程序通用选项

[*] Select only drivers that don't need compile-time external firmware

只显示那些不需要内核对外部设备的固件作 map 支持的驱动程序,除非你有某些怪异硬件,否则请选上

[*] Prevent firmware from being built

不编译固件.固件一般是随硬件的驱动程序提供的,仅在更新固件的时候才需要重新编译.建议选上

<*> Hotplug firmware loading support

加载热插拔固件支持 ， 在内核树之外编译的模块可能需要它

[ ] Driver Core verbose debug messages

让驱动程序核心在系统日志中产生冗长的调试信息,仅供调试

Memory Technology Devices (MTD) --->

特殊的存储技术装置,如常用于数码相机或嵌入式系统的闪存卡

Parallel port support --->

并口支持(传统的打印机接口)

Plug and Play support --->

即插即用支持,若未选则应当在 BIOS 中关闭"PnP OS".这里的选项与 PCI 设备无关

Block devices --->

ATA/ATAPI/MFM/RLL support --->

这个是有关各种接口的硬盘/光驱/磁带/软盘支持的，内容太多了，使用缺省的选项吧，如果你使用了比较特殊的设备，比如 PCMCIA 等，就到里面自己找相应的选项吧

SCSI device support --->

SCSI 设备

Multi-device support (RAID and LVM) --->

多设备支持(RAID 和 LVM).RAID 和 LVM 的功能是使多个物理设备组建成一个单独的逻辑磁盘

Fusion MPT device support --->

Fusion MPT 设备支持

IEEE 1394 (FireWire) support --->

IEEE 1394(火线)

I2O device support --->

I2O(智能 IO)设备使用专门的 I/O 处理器负责中断处理/缓冲存取/数据传输等烦琐任务以减少 CPU 占用,一般的主板上没这种东西

Networking support --->

网络支持

ISDN subsystem --->

综合业务数字网(Integrated Service Digital Network)

Input device support --->

输入设备

Character devices --->

字符设备

I2C support ---->

I2C 是 Philips 极力推动的微控制应用中使用的低速串行总线协议,可用于监控电压/风扇转速/温度等.SMBus(系统管理总线)是 I2C 的子集.除硬件传感器外"Video For Linux"也需要该模块的支持

Misc devices ---->

Multimedia devices ---->

多媒体设备

Graphics support ---->

图形设备/显卡支持

Sound ---->

声卡

USB support ---->

USB 支持

MMC/SD Card support ---->

MMC/SD 卡支持

第十二部分

File systems ----> 文件系统

<*> Second extended fs support

Ext2 文件系统是 Linux 的标准文件系统,擅长处理稀疏文件

[*] Ext2 extended attributes

Ext2 文件系统扩展属性(与 inode 关联的 name:value 对)支持

[*] Ext2 POSIX Access Control Lists

POSIX ACL(访问控制列表)支持,可以更精细的针对每个用户进行访问控制,需要外部库和程序的支持

[ ] Ext2 Security Labels

安全标签允许选择使用不同的安全模型实现(如 SELinux)的访问控制模型,如果你没有使用需要扩展属性的安全模型就别选

<*> Ext3 journalling file system support

Ext3 性能平庸,使用 journal 日志模式时数据完整性非常好(但怪异的是此时多线程并发读写速度却最快)

[*] Ext3 extended attributes

Ext3 文件系统扩展属性(与 inode 关联的 name:value 对)支持

[ ] Ext3 POSIX Access Control Lists

POSIX ACL(访问控制列表)支持,可以更精细的针对每个用户进行访问控制,需要外部库和程序的支持

[ ] Ext3 Security Labels

安全标签允许选择使用不同的安全模型实现(如 SELinux)的访问控制模型,如果你没有使用需要扩展属性的安全模型就别选

[ ] JBD (ext3) debugging support 仅供开发者使用

< > Reiserfs support

性能几乎全面超越 Ext2(处理稀疏文件比 Ext2 慢),小文件(小于 4k)性能非常突出,创建和删除文件速度最快,处理大量目录和文件(5k-20k)时仍然非常迅速.日志模式建议使用 Ordered,追求极速可使用 Writeback 模式,追求安全可使用 Journal 模式.建议使用 noatime,notail 选项挂载分区以提高速度和避免 bug.用于 NFS 和磁盘限额时需要额外的补丁

< > JFS filesystem support

IBM 的 JFS 文件系统

XFS support --->

碎片最少,多线程并发读写最佳,大文件(>64k)性能最佳,创建和删除文件速度较慢.由于 XFS 在内存中缓存尽可能多的数据且仅当内存不足时才会将数据刷到磁盘,所以应当仅在确保电力供应不会中断的情况下才使用 XFS

< > Minix fs support

老古董文件系统

< > ROM file system support

用于嵌入式系统的内存文件系统的支持

[ ] Quota support

磁盘配额支持,限制某个用户或者某组用户的磁盘占用空间,Ext2/Ext3/Reiserfs 都支持它

<*> Kernel automounter support

内核自动加载远程文件系统(v3,就算选也不选这个旧的)

< > Kernel automounter version 4 support (also supports v3)

新的(v4)的内核自动加载远程文件系统的支持,也支持 v3

CD-ROM/DVD Filesystems ---->

CD-ROM/DVD 文件系统

DOS/FAT/NT Filesystems ---->

DOS/Windows 的文件系统

Pseudo filesystems ---->

伪文件系统

Miscellaneous filesystems ---->

非主流的杂项文件系统

Network File Systems ---->

网络文件系统

Partition Types ---->

高级磁盘分区类型,不确定可以全不选

Native Language Support ---->

本地语言支持.如果你仅仅使用几种主流的 Linux 文件系统(ext2/3/4,Reiserfs,JFS,XFS),就不需要这个东西.但是如果你需要使用 FAT/NTFS 分区的话,就需要这个东西了.

第十三部分

Profiling support ---->

[ ] Profiling support (EXPERIMENTAL)

对系统的活动进行分析,仅供内核开发者使用

第十四部分

Kernel hacking ---> 内核 hack 选项，普通用户是用不着这个功能的

[ ] Show timing information on printks

在 printk 的输出中包含时间信息,可以用来分析内核启动过程各步骤所用时间

[*] Kernel debugging

不是内核开发者的别选

[ ] Magic SysRq key

不懂的千万别选

(14) Kernel log buffer size (16 => 64KB, 17 => 128KB)

[ ] Collect scheduler statistics

[ ] Debug memory allocations

[ ] Spinlock debugging

[ ] Sleep-inside-spinlock checking

[ ] kobject debugging

[ ] Compile the kernel with debug info

[ ] Debug Filesystem

[*] Verbose user fault messages

[ ] Wait queue debugging

[ ] Verbose kernel error messages

[*] Kernel low-level debugging functions

[ ] Kernel low-level debugging via EmbeddedICE DCC channel

[*] Kernel low-level debugging messages via S3C2410 UART

(0) S3C2410 UART to use for low-level debug

第十五部分

Security options --->安全选项，这里的选项不明白的建议不要选,否则有可能弄巧成拙.

[ ] Enable access key retention support

在内核中保留 authentication token 和 access key

[ ] Enable different security models

允许内核选择不同的安全模型,如果未选中则内核将使用默认的安全模型

第十六部分

Cryptographic options ---> 加密选项

[*[*] Cryptographic API

提供核心的加密 API 支持.这里的加密算法被广泛的应用于驱动程序通信协议等机制中.子选项可以全不选,内核中若有其他部分依赖它,会自动选上

[ ] HMAC support

为 IPSec 所必须,可为 PPPoE 提供压缩支持

< > Null algorithms

NULL 加密算法(什么也不做),用于 IPsec 协议的封装安全载荷模块(ESP)

< > MD4 digest algorithm

老旧的摘要算法,已经过时

<*> MD5 digest algorithm

主流摘要算法,128 位(已被中国山东大学王小云攻破,可以快速找到碰撞)

< > SHA1 digest algorithm

主流摘要算法,160 位(已被中国山东大学王小云攻破,可以快速找到碰撞),速度与 MD5 相当

< > SHA256 digest algorithm

更好的摘要算法,256 位,速度较 SHA1 稍慢

< > SHA384 and SHA512 digest algorithms

更好的摘要算法,384/512 位,速度大约只有 SHA1 的 40-50%

< > Whirlpool digest algorithms

最安全的摘要算法,512 位,已被列入 ISO 标准,目前最新版本为 3.0(2003 年发布)

< > Tiger digest algorithms

号称最快的摘要算法,192 位,专门为 64 位 CPU 进行了优化

<*> DES and Triple DES EDE cipher algorithms

老迈的(DES)和尚佳的(Triple DES)对称加密算法

< > Blowfish cipher algorithm

又老又慢的对称加密算法

<> Twofish cipher algorithm

很强的对称加密算法,使用较广

<> Serpent cipher algorithm

很强的对称加密算法

<> AES cipher algorithms

最佳的对称加密算法(Rijndael),128/192/256 位,强度最高,快速且节省内存

<> CAST5 (CAST-128) cipher algorithm

对称加密算法

<> CAST6 (CAST-256) cipher algorithm

对称加密算法

<> TEA and XTEA cipher algorithms

较弱的对称加密算法

<> ARC4 cipher algorithm

脆弱的流对称加密算法

<> Khazad cipher algorithm

对称加密算法

<> Anubis cipher algorithm

对称加密算法

<> Deflate compression algorithm

压缩算法,当在 IPSec 中使用 IPCOMP 协议时才需要

<> Michael MIC keyed digest algorithm

摘要算法,仅仅用于校验 iSCSI 设备传输的数据,因为算法本身比较脆弱

<> CRC32c CRC algorithm

摘要算法,可用于校验 iSCSI 设备传输的数据

<> Testing module

快速且丑陋的测试模块

Hardware crypto devices --->

仅有 VIA C7 系列处理器支持硬件加密(VIA PadLock 高级加密引擎)

第十七部分

Library routines ---> 库子程序

--- CRC-CCITT functions

传送 8-bit 字符,欧洲标准

--- CRC32 functions

用于点对点的同步数据传输中,传输网络数据包所必须的

<*> CRC32c (Castagnoli, et al) Cyclic Redundancy-Check

用于点对点的同步数据传输中,比如 iSCSI 设备

第十八部分

Load an Alternate Configuration File

读入一个外部配置文件

Save Configuration to an Alternate File

将配置保存到一个外部文件

飞凌嵌入式